НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки 6.050103 «Програмна інженерія»

на тему: «Веб-платформа для розміщення реклами. Серверна частина»

Виконав:
студент IV курсу, групи КП-52
Дяченко Дмитро Олександрович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.
Рибачок Н.А. _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай М.В. _____

Рецензент:
Доцент кафедри ЕІ ФЕЛ, к.т.н.,
Вунтесмері Ю.В. _____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки – 6.050103 «Програмна інженерія»

«ЗАТВЕРДЖУЮ»

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ
на дипломний проект студенту
Дяченко Дмитру Олександровичу

1. **Тема проекту:** «Веб-платформа для розміщення реклами. Серверна частина», керівник проекту Рибачок Наталія Антонівна, к.т.н., старший викладач, затверджена наказом по університету № 1331-С від «22» травня 2019 р.
2. **Термін подання** студентом завершеного проекту: «20» червня 2019 р.
3. **Вихідні дані для дипломного проектування:** див. Технічне завдання.
4. **Зміст пояснювальної записки:**
   – огляд існуючих програмних рішень;
   – обґрунтування вибору засобів реалізації;
   – опис розробленої системи;
   – аналіз розробленої системи.
5. **Перелік обов'язкового ілюстративного матеріалу:**
   – діаграма варіантів використання системи (креслення);
   – схема бази даних (креслення);
   – керівництво-користувача у вигляді блок схеми (плакат);
   – схема потоку даних та основних елементів системи (плакат);
6. **Консультанти розділів проекту**

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. **Дата видачі завдання:** «31» жовтня 2018 р.

КАЛЕНДАРНИЙ ПЛАН-ГРАФІК

№ з/п	Назва етапів роботи та питань, які мають бути розроблені відповідно до завдання	Термін виконання	Примітка
1.	Вивчення літератури за тематикою проекту	07.11.2018	
2.	Розроблення та узгодження технічного завдання	20.11.2018	
3.	Розроблення структури системи	10.12.2018	
4.	Підготовка матеріалів першого розділу дипломного проекту	21.12.2018	
5.	Розроблення архітектури серверної частини системи та планування бази даних	28.01.2019	
6.	Підготовка матеріалів другого розділу дипломного проекту	15.02.2019	
7.	Програмна реалізація системи	15.03.2019	
8.	Тестування системи	22.03.2019	
9.	Підготовка матеріалів третього та четвертого розділу дипломного проекту	15.04.2019	
11.	Оформлення документації дипломного проекту	01.06.2019	

Керівник дипломного проекту _____ Н.А. Рибачок

Студент _____ Д.О. Дяченко

АНОТАЦІЯ

Цей дипломний проект присвячений створенню програмного забезпечення серверної частини веб-платформи для розміщення реклами.

У роботі був проведений аналіз існуючих аналогів систем для розміщення реклами в мережі Інтернет. Були описані переваги та недоліки проаналізованих систем, а також функціональні можливості, що надаються користувачам.

За результатами аналізу було сформовано функціональні та нефункціональні вимоги до програмного забезпечення платформи, що розробляється. За результатами проведеного аналізу інструментів для розробки платформи було обрано відповідні технології. В дипломному проекті розроблено структуру бази даних, логіку, що описує взаємодію бази даних з серверною частиною. Також створено контролери, що відповідають за обробку інформації та реалізують функціональні можливості передбачені вимогами до програмного забезпечення платформи у повній мірі.

Архітектура системи дозволяє подальше її вдосконалення без внесення змін у структуру системи.

У результаті роботи над дипломним проектом розроблено веб-платформу, що дозволяє розміщувати рекламу у мережі Інтернет. Веб-платформа буде корисною для просування товарів та послуг у мережі Інтернет для користувачів.

ABSTRACT

This diploma project deals with the development of the server part of advertising web platform.

The analysis of existing analog system for advertising on the Internet was conducted in the project. The advantages and disadvantages were described as well as functionalities provided to users.

As results of the analysis functional and non-functional requirements for the software of the developed platform were formed. An analysis of the tools needed for developing a platform was conducted. Technologies for the development were choosen according to the analysis. An database structure and a logic, that describes the interaction of the database with the server was developed in the dilpoma project. Also server part controllers, that are responsible for information processing and implement all the functionalities required by the software platform to the fullest, were created.

The architecture of the system can be easily improved without changing the structure of the system if needed.

As a result of the diploma project web platform was developed, that allows people to advertise on the Internet. The web platform will be useful for promoting products and services on the Internet for users.

ДП.045440-01-90 Веб-платформа для розміщення реклами. Серверна частина. Відомість проекту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проекту		
ДП.045440-02-91	Веб-платформа для	5	
	розміщення реклами.		
	Серверна частина.		
	Технічне завдання		
ДП.045440-03-81	Веб-платформа для	5	
	розміщення реклами.		
	Серверна частина.		
	Пояснювальна записка		
ДП.045440-04-51	Веб-платформа для	4	
	розміщення реклами.		
	Серверна частина.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Веб-платформа для	1	
	розміщення реклами.		
	Серверна частина.		
	Керівництво користувача		
ДП.045440-06-99	Веб-платформа для	1	
	розміщення реклами.		
	Серверна частина.		
	Діаграма варіантів		
	використання системи		

Позначення	Найменування	Кіл-ть	Примітка
ДП.045440-07-99	Веб-платформа для	1	
	розміщення реклами.		
	Серверна частина.		
	Схема бази даних		
ДП.045440-08-98	Веб-платформа для	1	
	розміщення реклами.		
	Серверна частина.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-ПЛАТФОРМА ДЛЯ РОЗМІЩЕННЯ РЕКЛАМИ. СЕРВЕРНА ЧАСТИНА

Технічне завдання

ДП.045440-02-91

"ПОГОДЖЕНО"

Керівник проекту:

_____ Н.А. Рибачок

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.О. Дяченко

2018

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Веб-платформа для розміщення реклами. Серверна частина.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проектування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості веб-платформи для розміщення реклами з метою надання користувачеві можливості використання Інтернет-маркетингу для залучення та утримання клієнтів в мережі Інтернет.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Серверна частина веб-платформи повинна забезпечувати наступні основні функції:

– можливість реєстрації та авторизації користувачів;

– можливість створення рекламної кампанії та рекламного банера;

– можливість реєстрації сайту та створення рекламного блоку;

– можливість гнучкого налаштування таргетингу кампанії;

– можливість отримувати детальну статистику по кампаніям, банерам та блокам;

– можливість оновлення даних у базі даних;

– можливість швидко відповідати на запити.

Розробку виконати на платформі Node.js (мова програмування JavaScript) з використанням архітектурного шаблону клієнт-сервер.

Додаткові вимоги:

– наявність документації до програмного інтерфейсу платформи;
– наявність згоди на обробку персональних даних та користувацької згоди;
– дизайн сторінок з використанням білого, синього та темно-синього кольорів.

5. ВИМОГИ ДО ПРОЕКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проекту повинна бути розроблена наступна документація:

1) пояснювальна записка;
2) програма та методика тестування;
3) керівництво користувача;
4) креслення:
   – «Діаграма варіантів використання системи»;
   – «Схема бази даних»;

6. ЕТАПИ ПРОЕКТУВАННЯ

Вивчення літератури за тематикою роботи 02.11.2018
Розроблення та узгодження технічного завдання 21.11.2018
Розроблення структури системи 10.12.2018
Проектування бази даних 29.01.2019
Програмна реалізація системи 15.03.2019
Тестування системи..... 08.04.2019
Підготовка матеріалів текстової частини проекту 22.04.2019
Підготовка матеріалів графічної частини проекту..... 16.05.2019
Оформлення технічної документації проекту 27.05.2019

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-ПЛАТФОРМА ДЛЯ РОЗМІЩЕННЯ РЕКЛАМИ. СЕРВЕРНА ЧАСТИНА

Пояснювальна записка

ДП.045440-03-81

"ПОГОДЖЕНО"

Керівник проекту:

_____ Н.А. Рибачок

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.О. Дяченко

2019

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Веб-браузер – програмне забезпечення для під'єднаного до мережі Інтернет пристрою, що дає можливість взаємодії з даними на гіпертекстовій веб-сторінці.

Веб-фреймворк – каркас, призначений для створення динамічних веб-додатків та сервісів, що спрощує розробку та зменшує дублювання коду завдяки наявності готових компонентів.

Вебмайстер – користувач, який надає свій сайт для розміщення реклами на ньому.

Мультипарадигмальна мова програмування – мова програмування з підтримкою кількох стилів написання та організації структури програмного забезпечення.

Рекламодавець – користувач, який є замовником реклами для її розповсюдження.

Шаблонізатор – програмне забезпечення, що дозволяє використовувати HTML-шаблони для генерації кінцевих HTML-сторінок.

БД – база даних.

ПЗ – програмне забезпечення.

ПК – персональний комп’ютер.

ООП – об’єктно-орієнтоване програмування.

ОС – операційна система.

СУБД – система управління базами даних.

СКБД – система керування базами даних.

РК – рекламна кампанія, система запланованих заходів з просування та реклами послуги чи товару.

CPC – cost per Click (ціна за 1 натискання).

CPM – cost per mille (ціна за 1000 показів).

CTR – click-through rate, показни клікабельності.

CPA – cost per action (ціна за дію).

JVW – Java Virtual Machine (віртуальна машина Джава).

JSON – JavaScript Object Notation (текстовий формат для обміну даними, оснований на JavaScript, може бути прочитаним людиною).

REST – Representational State Transfer, підхід до архітектури мережевих протоколів, які забезпечують доступ до інформаційних ресурсів.

ER – entity relationship, модель “сутність-звязок”.

CSRF – Cross Site Request Forgery, міжсайтова підробка запиту.

API – Application Programming Interface (прикладний програмний інтерфейс).

Bid – ціна попиту, найвища ціна за якою покупець згоден купити товар або послугу.

Coroutine – співпрограма або співпроцедура.

GUI – Graphical User Interface (графічний інтерфейс користувача).

HTML – HyperText Markup Language (мова розмітки гіпертексту).

HTTP – HyperText Transfer Protocol (протокол передачі гіпертексту).

PyPI – Python Package Index (каталог пакетів Python).

REST – Representational State Transfer (передача стану подання).

UML – Unified Modeling Language (уніфікована мова моделювання).

URL – Uniform Resource Locator (уніфікована адреса ресурсу).

Unicode – стандарт кодування символів.

ORM – Object-Relational Mapping, технологія програмування, яка пов’язує бази даних з концепцією об’єктно-орієнтованих мов програмування, створюючи віртуальну об’єктну базу.

Perl – високорівнева, інтерпретована, динамічна мова програмування загального призначення.

ECMAScript – стандарт мови програмування, затверджений міжнародною організацією ECMA згідно зі специфікацією ECMA-262.

ВСТУП

Як відомо, одним із найважливіших факторів успіху продажу та просування товарів чи послуг є їх реклама. Реклама несе в собі інформацію в зрозумілій і стислій формі, що доводить до уваги потенційних кінцевих споживачів найбільш важливі факти про продукт, який рекламується. Реклама відіграє одну з найважливіших ролей в реалізації маркетингових стратегії. Вона привносить соціально-культурний і психологічний вплив на суспільство.

Використання засобів реклами в інтернеті, фактично новій сфері, де люди проводять дуже багато часу, є надзвичайно ефективним рішенням через велике охоплення аудиторії. Також, перевагою Інтернет-реклами є дешевизна. Розмістити рекламу на сайті набагато дешевше ніж опублікувати її в якомусь журналі та в сотні разів дешевше, ніж трансляція реклами на телебаченні. Так як якісна Інтернет-реклама не повинна бути нав’язливою, фахівці погоджуються, що простота є візиткою сучасної реклами.

За допомогою Інтернет-реклами освоюються нові ринки збуту. Являючись засобом конкурентної боротьби, реклама загострює її, що сприяє підвищенню якості обслуговування. Реклама забезпечує можливість збільшення об'ємів продажів. Особливого значення реклама набуває в умовах розвитку інформаційного суспільства. Вона стає дуже важливим і незамінним інструментом маркетингу, що встановлює, підтримує і розвиває комунікації між підприємством і споживачами. За допомогою реклами підтримується зворотний зв'язок з ринком. Використовуючи можливості дії реклами на споживача, вона не лише сприяє формуванню попиту, але і керує ним. Реклама не може і не повинна компенсувати не високий рівень якості продуктів і обслуговування клієнтів. Вона служить лише засобом доведення до споживачів інформації про продукти і послуги високої якості. У цьому і полягає актуальність даної теми.

Даний дипломний проект присвячено створенню веб-платформи для розміщення реклами товарів або послуг.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Загальний опис проблеми Інтернет-реклами

Сьогоденні тенденції розвитку ринку сприяють сильному розвитку новітніх підходів до позиціонування та просування товарів і послуг на ринку. В попередні роки активного розвитку набула Інтернет-реклама, перевагою якої є отримання найбільшого ефекту від потенційної цільової аудиторії. Актуальність інтернет-реклами обумовлена тим, що кількість користувачів Інтернету в Україні складає 58% станом на 2018 рік.

Інтернет-реклама являє собою великий комплекс заходів, виконання яких не завжди спрямоване на активну пропозицію товару або послуги. Інтернет-просування – це процес використання практичних заходів в мережі Інтернет, основною метою яких є результативна популяризація певних інформаційних одиниць (сайт, ідея, об’єкти інтелектуальної власності, фотографії, відеоролики, товари, послуги та інше).

Під товаром, в свою чергу, розуміється як все, що може бути цінністю для споживачів кінцевого ринку, причому ця цінність може бути властива матеріальному об’єкту, послузі, компанії, місцю, людині або ідеї.

У сучасних умовах багато підприємств закриваються, розвивається малий і середній бізнес та торгівля через Інтернет, що сприяє розвитку Інтернет-реклами. Також їй властива цінова різноманітність, яка залежить від місця розміщення реклами, але загалом реклама в Інтернеті має меншу вартість на відміну від більшості видів медіа реклами.

Але наразі великі рекламні платформи, такі як Google Ads, Facebook, Youtube та багато інших, мають деякі обмеження на рекламу товарів та послуг, які належить до певних категорій. Наприклад, заборонено рекламування криптовалют та всього що, пов’язано з ними.

Криптовалюта – це цифрова валюта, що функціонує завдяки механізму асиметричного шифрування. У світі налічується близько тисячі криптовалют, але найвідомішою є Bitcoin (біткойн). В Україні останнім

часом підвищилася увага до біткойнів через стрімке зростання їхніх цін. Це новий вид активів, який відрізняється від звичайних для нас грошей або інших фінансових активів. Від грошей відрізняється тим, що ви володієте валютою безпосередньо і для цього непотрібний банк чи якийсь дозвільний орган. Зберігається криптовалюта в особистому електронному гаманці. Одиниця криптовалюти – це код, який народжується в результаті складних комп'ютерних математичних обчислень.

Ті сервіси, які дозволяють рекламувати криптовалюти або товари чи послуги з інших заборонених груп, є набагато менш якісними і мають набагато менше можливостей для гнучкого налаштування Інтернет реклами. Тому на сьогоднішній день заповнення цієї ніші Інтернет-реклами є актуальним.

1.2. Аналіз існуючих програмних рішень

1.2.1. *Google Ads (Google AdWords)*

Google Ads – це сервіс контекстної реклами із зручним інтерфейсом (рис 1.1 та 1.2), який надає багато інструментів для створення рекламних банерів. При цьому Google Ads підходить для будь-якого типу компаній: як для великих холдингів, так і для маленьких фірм. Ця технологія інтернет-маркетингу надає право на розміщення реклами на сторінках результатів запитів пошуку Google. Найбільша з переваг сервісу – це гнучкість системи: можливість встановити бюджет на деякий срок, обмежити демонстрацію реклами географічними або часовими рамками, додавати та змінювати ключові слова, або навіть змінювати зміст самої реклами. Також Google Ads працює за системою pay-per-click, тобто рекламодавець платить лише за тих клієнтів, що зацікавилися рекламою та перейшли за посилання реклами.

Принцип роботи сервісу нескладний. Клієнт створює рекламний банер, що має містити в собі ключові слова, які мають відношення до товару або послуги. Коли потенційні клієнти будуть робити запити, які міститимуть в собі ключові слова, система покаже їм рекламний банер

поряд з результатами пошуку і клієнту залишиться лише натиснути на посилання.

Google Ads – один з найбільш ефективних рекламних інструментів. Найбільша перевага перед іншими рекламними платформами – таргетинг.

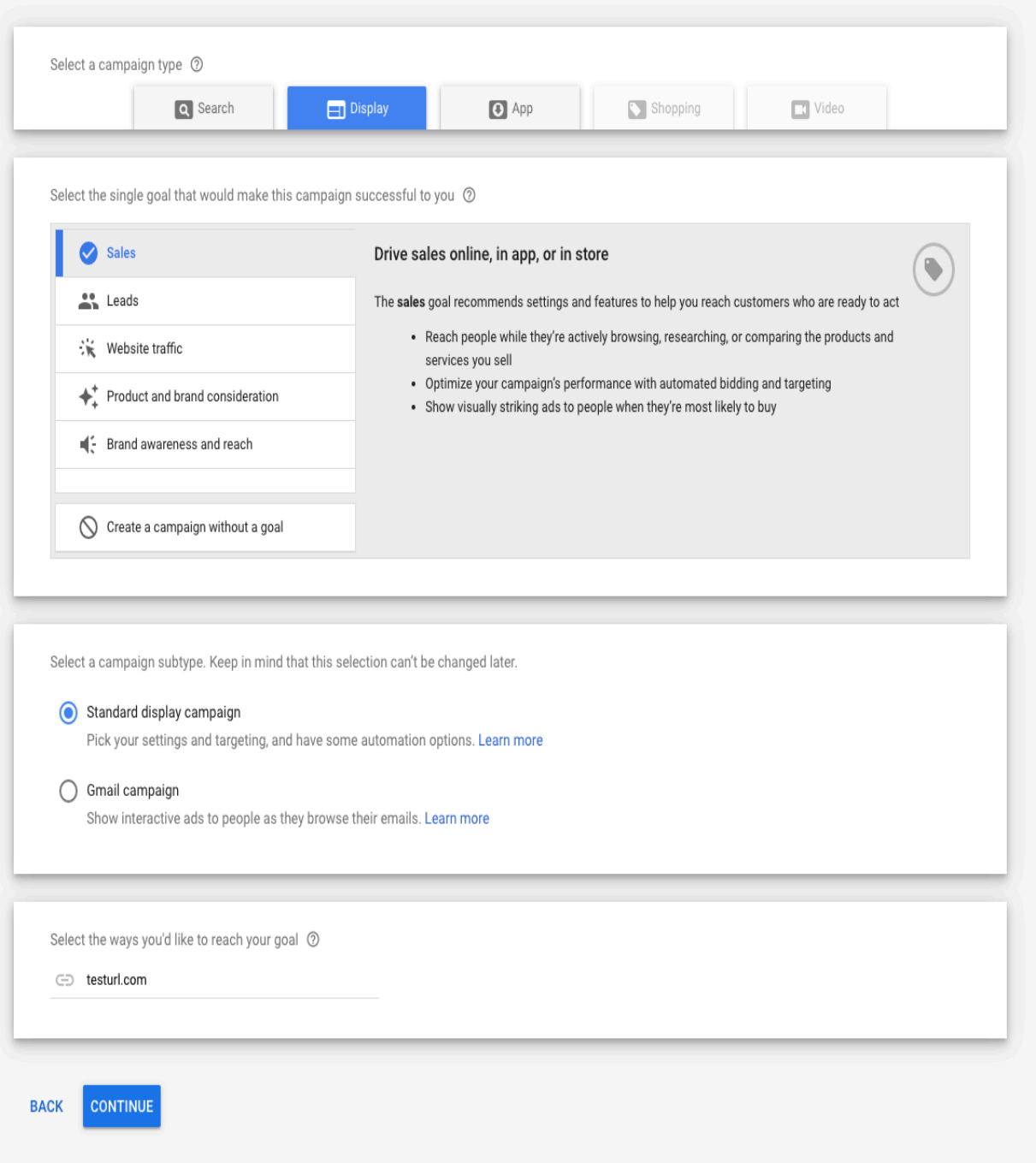


Рис. 1.1. Створення кампанії Google Ads

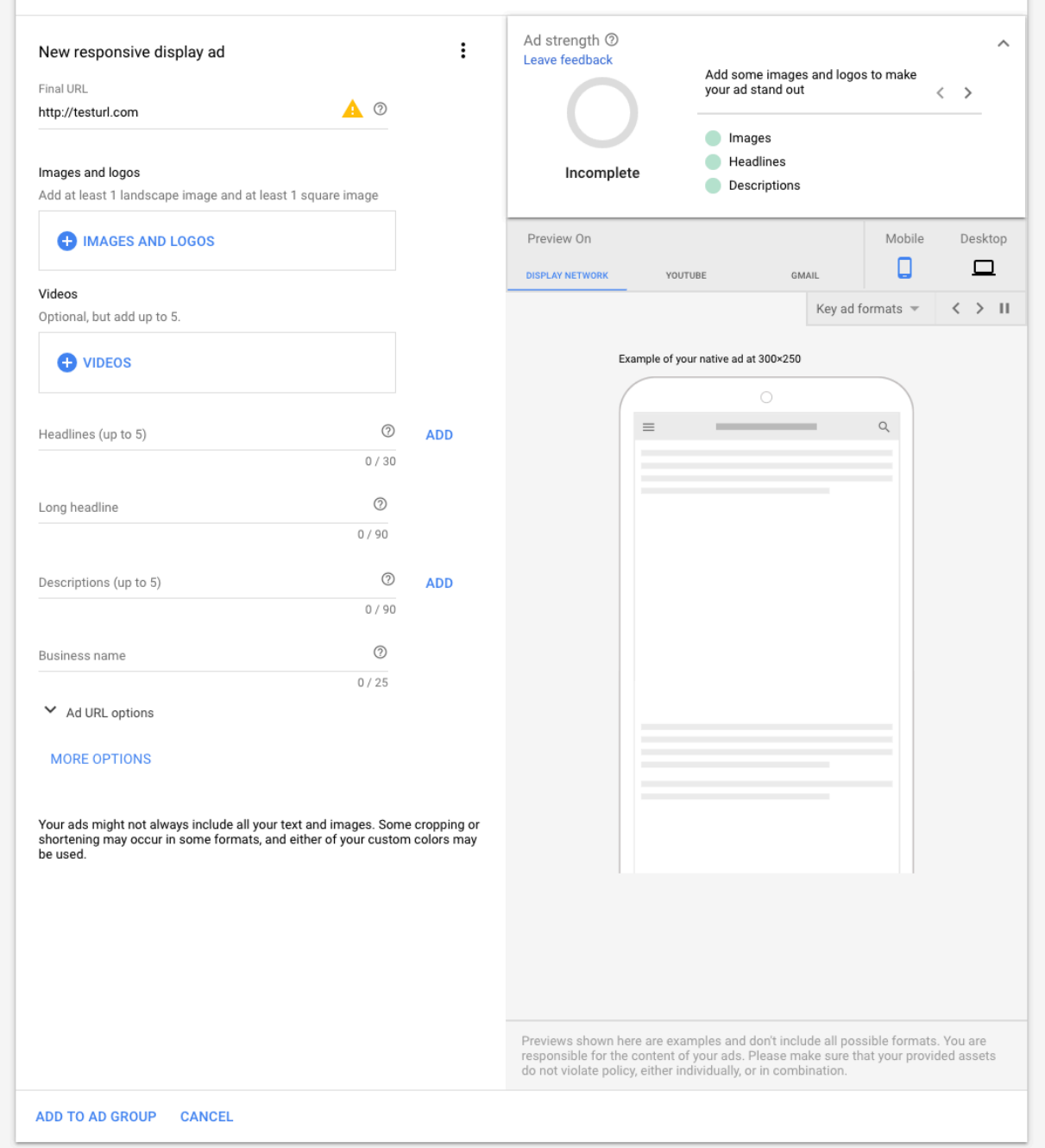


Рис. 1.2. Створення адаптивного рекламного банера

Недоліком цієї платформи є заборона на рекламування деяких категорій товарів та послуг, таких як:

- підроблені товари;
- небезпечні товари та послуг;
- великі обмеження на рекламу криптовалют.

1.2.2. *Яндекс.Діректом*

Яндекс.Діректом – це рекламна система, за допомогою якої користувач може розміщувати контекстні оголошення (рекламу) на сторінках Яндекс.Пошуку і на партнерських сайтах рекламної мережі. Реклама показуються виключно тим споживачам, які вже зайняті пошуком схожих товарів чи послуг на Яндексі або інших сайтах. Розміщуючи рекламу в Яндекс.Діректом, користувач платить за натискання по рекламі, тобто рекламна кампанія працює за принципом CPC кампанії.

Яндекс.Діректом має ряд переваг у порівнянні з іншими рекламними каналами. У більшості випадків, при гарному налаштуванні реклами, вона принесе нових клієнтів. Контекст пропонує споживачам те, що вони шукають. Реклама в Яндекс.Діректом привертає увагу “потрібних споживачів”, але при цьому вона не є нав’язливою і не викликає роздратування у інших.

Так як реклама в Яндекс.Діректом також працює за ключовими словами, користувач може пропонувати свої товари або послуги аудиторії, яка зацікавлена в цьому.

Система Яндекс.Діректом доволі гнучка. Користувач має можливість редагувати рекламу, коли захоче, при цьому змінювати тексти, зображення, додавати нові налаштування, а також тестувати різноманітні варіанти реклами.

Також, варто відзначити, що система має налаштування ретаргетингу. Ретаргетинг – це технологія рекламування товарів або послуг тим споживачам, які вже були на вашому сайті і здійснили на ньому якісь дії. Користувач сам може налаштувати, якім споживачам показувати рекламу. Наприклад, можна обрати для показу реклами користувачів, які пробули на сайті більше хвилини, але так і не здійснили покупку.

Як і усі великі рекламні системи, Яндекс.Діректом має ряд обмежень на рекламу товарів та послуг з деяких категорій, наприклад криптовалюти.

1.2.3. *AdBean*

AdBean – це сучасна рекламна платформа, яка об'єднує в собі багато переваг як класичних тізерних мереж, так і CPA партнерських програм. Для кожного вебмайстра в системі існує можливість обирати і підключати лише ті рекламні кампанії, які будуть найбільш ефективні та принесуть високий дохід рекламній площадці.

До переваг цього сервісу можна віднести:

- якісна підтримка (підтримка користувачів 24/7);
- швидка модерація (підтвердження сайтів триває не більше години);
- підвищення ставок (для сайтів з високою оцінкою роздаються бонуси);
- висока реферальная ставка 10%.

До недоліків можна віднести:

- високий поріг входу, тобто, сайт повинен мати відвідуваність мінімум 500 унікальних користувачів в день;
- у системі можуть приймати участь лише російські або українські сайти;
- потрібно обов'язково мати на сайті скрипти Яндекс.Метрика, Google Analytics або Top.Mail.ru.

AdBean також легкий у використанні. Користувачу потрібно зареєструватися, після чого користувач отримує можливість увійти до мережі та створити свою РК. Або ж, якщо користувач хоче стати вебмайстром, є можливість подати заяву для розгляду власного сайту. Після додавання сайту з'являється можливість створити рекламний блок і розмістити його на своєму сайті.

Ця рекламна мережа має досить обширні налаштування таргетингу для реклами, але все ж таки досить сильно поступається у своїй функціональності та гнучкості налаштувань Google Ads.

1.2.4. *Coinzilla*

Coinzilla – відносна молода рекламна мережа, започаткована у 2017 році, яка створює комплексні рішення для максимально ефективної монетизації сайтів з криптовалютною тематикою.

Особливістю системи є те, що всі платежі проводяться у цифровій валюті – Bitcoin. Проект забезпечує своїх користувачів щоденними виплатами, захистом від шахраїв та прозорою статистикою, що оновлюється в режимі реального часу.

Інтерфейс системи повністю представлений на англійській мові та має гарне дизайнерське оформлення. На головній сторінці присутня уся необхідна інформація, яка може зацікавити рекламодавців, так і власників інтернет-ресурсів, тобто вебмайстрів, з криптовалютною тематикою зі всього світу.

Система має дуже легкий і швидкий процес проходження реєстрації користувача. Сервіс пропонує декілька різних форматів реклами для рекламодавців, а також доволі гнучкі налаштування таргетингу реклами. Також простим є процес реєстрації сайту та створення рекламного блоку для вебмайстра.

На основі проведеного аналізу було виділено такі переваги сервісу:

- реферальная система;
- зручність та легкість у використанні;
- швидкі платежі;
- швидке підтвердження нових кампаній;
- зручна та прозора система звітності (статистика).

До недоліків можна віднести:

- відсутність української та російської версії сайту;
- дуже високий мінімальний депозит;
- дуже жорсткі критерії для реєстрації сайту.

1.2.5. *Висновки*

Отже, було розглянуто існуючі реалізації систем розміщення Інтернет-реклами та визначено їх переваги та недоліки. Найбільшим недоліком великих та популярних систем є заборона на рекламування певних категорій контенту (криптовалюти і т.п.), але найбільшою перевагою – висока гнучкість налаштування таргетингу реклами, але також, завдяки такій гнучкості, процес налаштування РК може бути дуже складним, тому звичайному користувача може знадобитися допомога спеціалістів з налаштування таргетингу реклами у певній рекламній мережі. Найбільшою перевагою маленьких, спеціалізованих на конкретній рекламі мереж, є дозволи на рекламування криптовалют та інших категорій товарів та послуг, які заборонені відповідними сервісами Google або Яндекс. При цьому налаштування таргетингу кампанії є менш гнучким, але при цьому воно також є простішим. Недоліком такої платформи є її вузька спеціалізація, тобто, або платформа спрямована лише на російські сайти, або лише на криптовалюти та інше.

На основі аналізу існуючих програмних рішень було сформовано список вимог до розроблюваної системи, які детальніше описано в підрозділі 3.1.

1.3. Узагальнення існуючих функцій аналогів

Під час процесу аналізу існуючих програмних рішень та вимог до розроблюваної системи було виділено такі функціональні вимоги:

- необхідність авторизації для отримання доступу до функцій веб-платформи. Внутрішнє розділення ролей користувачів на Рекламодавців (Advertiser) та Вебмайстрів (Publisher);
- користувач, повинен мати можливість поповнення балансу свого акаунта;

- забезпечити можливістю користувача на подання заяви для отримання статусу Вебмайстер. Кожен користувач автоматично стає Рекламодавцем після завершення процесу реєстрації;
- користувач повинен мати можливість заповнити особисті дані на сторінці особистого кабінету;
- користувач, який має статус Рекламодавець, повинен мати можливість створити рекламну кампанію (далі РК);
- користувач, який має статус Рекламодавець, повинен мати можливість гнучкого налаштування рекламної кампанії;
- користувач, який має статус Рекламодавець, повинен мати можливість створити рекламний банер (далі реклама) в РК;
- користувач, який має статус Рекламодавець, повинен мати можливість перегляду статистики реклами та РК на відповідних сторінках;
- користувач, який має статус Рекламодавець, повинен мати можливість виставляти ціну на рекламу;
- користувач, який має статус Рекламодавець, повинен мати можливість поповнювати свої РК;
- користувач, який має статус Вебмайстер, повинен мати можливість зареєструвати сайт для розміщення на ньому рекламних блоків;
- користувач, який має статус Вебмайстер, повинен мати можливість створювати рекламні блоки для розміщення їх на сайтах;
- користувач, який має статус Вебмайстер, повинен мати можливість переглядати статистику по кожному рекламному блоку окремо, та загальну статистику.

На основі переліку функціональних вимог було побудовано UML діаграму прецедентів (див. Додатки). Також було визначено нефункціональні вимоги:

- підтримка роботи платформи у найбільш поширених веб-браузерах (Chrome, Opera, Safari, Firefox) та у мобільних браузерах;
- зручність, зрозумілість та простота інтерфейсу користувача;
- наявність англійської локалізації.

2. ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1. Обґрунтування вибору типу програмного забезпечення

Було розглянуто три можливі варіанти розробки платформи для розміщення реклами та особливості їх реалізації.

- Веб-платформа
  - відсутність апаратних вимог до клієнта, потрібність лише в сучасному веб-браузері;
  - не вимагає встановлення та обслуговування;
  - найдешевша розробка.
- Десктопний додаток
  - розроблення для конкретної ОС, апаратні вимоги до клієнта;
  - вимагає встановлення та обслуговування;
  - необхідні витрати на розробку для кожної ОС.
- Мобільний додаток
  - розроблення для конкретної платформи;
  - вимагає встановлення та обслуговування;
  - необхідні витрати на розробку для кожної платформи.

Враховуючи переваги та недоліки цих варіантів розробки, а також результати першого розділу дипломного проекту з аналізу існуючих рішень, було вирішено розробляти веб-платформу для розміщення реклами. Перевагами такої реалізації буде незалежність від платформи користувача, доступність з будь-якого пристрою, на якому є браузер та підключення до мережі Інтернет.

2.2. Вибір мови програмування для розробки серверної частини веб-платформи

Під час першого етапу розробки програмного проекту розробник має провести дослідження та запланувати які мови програмування і технології є доцільними для розроблення продукту. Серверна частина веб-платформи

складається із коду, що обробляє дані клієнта, працює з базами даних, виконує основну логіку програми.

Для реалізації цієї частини веб-платформи було вирішено обрати мову програмування, яка задовольняє таким вимогам: мультипарадигмальність, інтерпретованість, підтримка ООП, великий об’ємом стандартної бібліотеки та наявність вибору фреймворків.

2.2.1. *Мова JavaScript*

JavaScript – це об’єктно-орієнтована, прототипна та динамічна мова програмування, реалізація стандарту ECMAScript. Цю мову класифікують як прототипну (підмножина об’єктно-орієнтованої), скриптову з динамічною типізацією. Вона також частково підтримує інші парадигми програмування і деякі відповідні архітектурні властивості, такі як: динамічна та слабка типізація, автоматичне керування пам’яттю, функції як об’єкти першого класу, прототипне наслідування.

JavaScript має багато властивостей, притаманних об'єктно-орієнтованим мовам, але завдяки концепції прототипів підтримка об'єктів в ній відрізняється від традиційних мов ООП. Крім того, JavaScript має ряд властивостей, притаманних функціональним мовам: функції, як об'єкти першого класу, об'єкти, як списки, анонімні функції, замикання (closures), що додає мові додаткову гнучкість. JavaScript має C-подібний синтаксис.

Для розробки сервера на мові JavaScript потрібне використання спеціальних рушіїв:

Rhino – рушій від компанії Mozilla, який написаний на мові Java. Основним плюсом цього рушія є те, що він написаний поверх стандартної JVM, це означає що його можна використовувати в будь-якому середовищі, де працює Java.

SpiderMonkey – ще один рушій від компанії Mozilla, що написаний на C. Він в основному використовується в ПЗ, яке написане на C/C++ так потребує скриптову мову.

V8 – рушій від Google, який використовується в Chrome. На сьогоднішній день це найкрутіший, швидкий і потужний рушій, в якому JS-код напряму перетворюється в ассемблерний код, що дозволяє обійти по швидкості усі інші рушії. На базі цього рушія побудована найпопулярніша серверна платформа – Node.js.

Саме поява Node.js перетворила JavaScript на мову загального використання з можливістю виконувати JavaScript скрипти на сервері та відправляти користувачам результати їх виконання.

Сьогодні мова JavaScript є однією з найбільш популярних. Але асинхронність мови в поєднанні з однопоточністю платформи Node.js сприяє зниженню обчислювальної продуктивності.

2.2.2. *Мова PHP*

PHP – мова, у код якої можна вбудовувати безпосередньо html-код сторінок, які, у свою чергу, коректно оброблюватимуться PHP-інтерпретатором. Перед тим як відправити сторінку, код PHP виконується на сервері і браузеру віддається результат у вигляді HTML-сторінки, яка може сильно відрізнятись від тієї яка присутня на сервері. PHP називають мовою серверних скриптів, на відміну від JavaScript. Це означає, що PHP–скрипт виконується на сервері, а клієнту передається результат його роботи, тоді як JavaScript повністю передається на клієнтську машину і там виконується браузером.

Стандартний інтерпретатор мови PHP вільно розповсюджується. Він може бути розгорнутий на більшості веб-серверів практично на будь-якій операційній системі та платформі безкоштовно.

Синтаксис PHP подібний до синтаксису мови C та містить деякі запозичені конструкції з Perl (наприклад, асоціативні масиви). В процесі історичного розвитку мова розширювалась та доповнювалась новими і запозиченими з інших мов можливостями без встановлення послідовних правил та конвенцій. З одного боку, це спричинило неузгодженість

синтаксису PHP, але з іншого боку, мова отримала не високий поріг входження для програмістів з різних галузей, обумовлений наявністю традиційних та знайомих конструкцій.

PHP є мультипарадигмальною мовою програмування, яка підтримує процедурну, функціональну та об'єктно-орієнтовану парадигми. Починаючи лише з п’ятої версії мова підтримує ООП, інтерфейси, абстрактні класи та методи, проте множинне наслідування не підтримується.

Інтерпретатор PHP забезпечує обробку скриптів зі збереженням кросплатформності розроблюваних додатків. Швидкодія додатків може бути збільшена з допомогою спеціального програмного забезпечення – акселераторів. Програміст не має піклуватися про розподіл і звільнення пам'яті, тому що ядро PHP надає засоби для автоматичного керування пам'яттю – вся виділена пам'ять повертається в систему після завершення роботи скрипта.

Можна виділити такі особливості мови:

- наявність інтерфейсів до великої кількості баз даних, в PHP є вбудовані бібліотеки для роботи з PostgreSQL, MySQL, Oracle, sybase та іншими;
- наявність сирцевого коду;
- безкоштовність;
- ефективність.

Важливою перевагою PHP є те, що вона належить до інтерпретованих. Це дозволяє обробляти сценарії з достатньо високою швидкістю.

До недоліків можна віднести:

- непередбачуваність нових версій;
- не підтримує Unicode (у версіях до 6.0);
- незручний дизайн мови.

Найбільш поширені PHP-фреймворками для розробки веб-додатків є Laravel, CodeIgniter, Symfony, Zend.

2.2.3. *Мова Python*

Python – інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Python підтримує модулі та модульні пакети, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, а саме: об'єктно-орієнтована, функціональна, процедурна та аспектно-орієнтована.

До основних переваг можна віднести:

- чистий синтаксис;
- переносимість програм (що властиве більшості інтерпретованих мов);
- стандартний дистрибутив має велику кількість корисних модулів;
- дуже зручний для розв'язання математичних проблем;
- відкритий код.

Python має ефективні структури даних високого рівня та простий, але ефективний підхід до об'єктно-орієнтованого програмування. Елегантний синтаксис Python, динамічна обробка типів, а також те, що це інтерпретована мова, роблять її ідеальною для написання скриптів та швидкої розробки прикладних програм у багатьох галузях на більшості платформ. Але Python, як і багато інших інтерпретованих мов, які не застосовують, наприклад, JIT-компілятори, мають загальний недолік – порівняно не високу швидкість виконання програм.

Портованість мови дозволяє працювати на будь-якій з відомих платформ, втім, на відміну від інших портованих систем, Python надає підтримку характерних для платформи технологій. Існує спеціальна версія мови для віртуальної машини Java – Jython, рішення для інтеграції з платформою Microsoft .NET – IronPython, Python.Net.

Всі дані в програмах, написаних на Python, є об’єктами, в тому числі функції, методи, класи та модулі. Мова містить як примітивні типи, так і вбудовані колекції – списки, кортежі, словники, множини.

Характерними для мови є чіткість та послідовність синтаксису, модульність та розширюваність. Модулі Python об’єднуються в пакети, формуючи розроблені додатки чи бібліотеки. Для отримання доступу до простору імен модуля використовується оператор підключення import.

Веб-фреймворки, що використовуються у мові Python, у свою чергу, не підтримують асинхронне програмування за замовчуванням, але підтримують корутини (coroutines), за допомогою яких є можливість домогтись асинхронної обробки запитів. Таким чином, мова програмування Python має повний інструментарій для досягнення необхідної масштабованості веб-додатку.

2.3. Вибір фреймворку для розробки серверної частини веб-платформи

Розглянемо два фреймворки Express.js та Sails.js. Express – найгнучкіший, простий та швидкий. Sails – створений для швидкої розробки веб-додатків на принципах Ruby on Rails та Express.

2.3.1. *Experess.js*

Почнемо з найпростішого фреймворка, який використовується на платформі Node.js. Express використовується для розробки додатків вже досить довго і завдяки своїй стабільності міцно займає позицію одного з найбільш популярних фреймворків. Основна особливість цього фреймворка полягає в тому, що для нього характерний невеликий обсяг базової функціональності. Всі інші функції потрібно буде використовувати з зовнішніх модулів. По суті, Express.js в чистому вигляді – це сервер і у нього може не бути жодного модуля. Завдяки цьому мінімалізму розробник отримує легкий і швидкий інструмент, який він може розширювати. В

результаті, цей фреймворк забезпечує розробнику можливість вирішувати будь-які завдання, не обмежуючи його у виборі засобів.

З одного боку відсутність готових універсальних рішень фактично означає, що кожний створений додаток буде унікальним. З іншого боку, розробнику потрібно самостійно підбирати і організовувати модулі, а це передбачає великий обсяг роботи і, відповідно, вимагає від розробника більше часу і зусиль.

Переваги:

- простота;
- гнучкість;
- швидкість;
- масштабованість;
- детальна документація;
- великий вибір модулів;
- велике співтовариство.

Недоліки:

- великий обсяг ручної роботи.

Express.js підходить для:

- початківців програмістів, які націлені на професійний ріст в Node.js;
- великих проектів, які передбачають кастомізацію;
- випадків, коли необхідна довгострокова підтримка програми.

2.3.2. *Sails.js*

Цей фреймворк розроблений як повний готовий продукт, який вже включає в себе достатню функціональність для того, щоб можна було почати роботу, і при цьому використовує мінімальну кількість зовнішніх модулів. Відповідно, такий фреймворк спочатку буде важче, тобто матиме набагато більше зайвих функцій, ніж Express.js.

З одного боку, від розробника потрібна мінімальна кількість зусиль, так як для створення веб-додатка використовується власна функціональність фреймворка. Немає необхідності вникати в тонкощі процесу – можна просто взяти готове перевірене рішення. З іншого боку, розробка програми буде обмежена рамками наявних можливостей фреймворка, так як зовнішніх модулів для Sails.js набагато менше, ніж для Express.js.

Відмінною особливістю фреймворка є вбудована технологія програмування Waterline ORM (Object-Relational Mapping), яка використовується для забезпечення зв'язку з різними базами даних. Наявність такого компонента можна було б віднести до переваг, але в процесі роботи можна наштовхнутися на певні обмеження.

Переваги:

- багата власна функціональність;
- документація в одному місці.

Недоліки:

- важкий;
- повільний;
- обмеження Waterline;
- недостатньо детальна документація.

Sails.js підходить для:

- швидкого старту проекту;
- швидких стартапів, які не передбачають розширення в майбутньому;
- додатків реального часу, де потрібно моментальна реакція;
- початківців Node.js програмістів;
- додатків, що не вимагають довгострокової підтримки.

2.3.3. *Висновкни*

Отже, підсумовуючи, можна сказати, що Express.js більш простий і легкий у використання та набагато гнучкіший у налаштуванні, а також підходить для довгострокової підтримки, тому для розроблення серверної частини буде використовувати Express.js.

2.4. Вибір системи керування базами даних

СКБД для веб-платформи повинна відповідати таким встановленим критеріям:

- висока швидкодія;
- безпека ;
- підтримка індексів для підвищення ефективності виконання запитів;
- легкість та зручність використання;
- відкритий код.

Було розглянуто такі бази даних: MySQL, MongoDB.

2.4.1. *MySQL*

MySQL – система керування реляційними базами даних. Зараз MySQL – одна з найпоширеніших СКБД. Вона використовується для створення динамічних веб-сторінок, оскільки має чудову підтримку з боку різноманітних мов програмування. MySQL надає багатий набір функціональних можливостей, які підтримують безпечне середовище для зберігання, обслуговування і отримання даних. MySQL – характеризується великою швидкістю, стійкістю і простотою використання, була розроблена для підвищення швидкодії обробки великих баз даних.

Можливості сервера MySQL: простота у встановленні та використанні, підтримується необмежена кількість користувачів, що одночасно працюють із БД, кількість рядків у таблицях може досягати 50 млн, висока швидкість виконання команд; наявність простої і ефективної системи безпеки.

Серед основних переваг MySQL відзначають наступні:

- масштабованість;
- переносність;
- зв’зність;
- безпека;
- швидкість;
- зручність;
- відкритий код.

2.4.2. *MongoDB*

MongoDB – документо-орієнтована система керування базами даних з відкритим вихідним кодом. Вона не потребує опису схеми таблиць. MongoDB займає нішу між швидкими і масштабованими системами, що оперують даними у форматі ключ/значення і реляційними СКБД.

MongoDB підтримує зберігання документів в JSON-подібному форматі, має гнучку мову для формування запитів, повністю підтримує індекси, підтримує журналювання операцій з даними в БД, також підтримує реплікацію. У MongoDB є вбудовані засоби із забезпечення шардінгу. MongoDB підтримується багатьма мовами програмування, легка у використанні та налаштуванні.

Основні переваги MongoDB:

- відсутність схеми таблиць;
- легка масштабованість;
- швидкість;
- збереження даних у вигляді JSON-об’єктів;
- безпека;
- журналювання;
- відкритий код;
- вичерпна документація.

Для веб-платформи для розміщення реклами може підійти як реляційна база даних MySQL так і документо-орієнтована MongoDB. Було вирішено використовувати MongoDB, однією з основних переваг якої є формат документів – JSON, це заощадить час під час роботи з нею. Відсутність схеми таблиць полегшить та пришвидшить проектування бази, та дасть можливість легко і швидко модифікувати колекції при необхідності. Також MongoDB легко масштабується при потребі та зовсім не поступається у швидкодії. Великий досвід роботи автора з цією СКБД теж вплинув на вибір.

2.5. Висновки

У даному розділі було розглянуто та проаналізовано основні інструменти для розробки веб-платформ. Після аналізу інформації було обрано конкретні засоби для реалізації веб-платформи для розміщення реклами.

Для розробки веб-платформи було вирішено обрати мову програмування JavaScript, оскільки:

- мова Python показує низьку швидкодію, також чистий синтаксис не є дуже зручним;
- дизайн мови PHP є досить незручним ;
- автор має досить великий досвід використання мови JavaScript, а саме платформи Node.js з фреймворком Express.js.

В якості системи керування базами даних було вирішено обрати MongoDB, оскільки вона є дуже простою у використанні разом з Node.js та Express, має всі необхідні функції та показує високу швидкодію виконання команд.

3. ОПИС РОЗРОБЛЕНОЇ СИСТЕМИ

3.1. Функціональні вимоги до системи

Неавторизовані користувачі мають такі можливості:

- увійти до системи;
- зареєструватися у системі.

Зареєстрований та авторизований у системі користувач, який має роль рекламодавця, має такі можливості:

- заповнення особистої інформації користувача;
- поповнення балансу користувача;
- створення РК;
- поповнення балансу кампанії;
- створення рекламного банера;
- редагування рекламного банера;
- регулювання ціни рекламних банерів;
- гнучке налаштування таргетингу кампанії;
- запуск РК;
- запуск окремого банера;
- перегляд статистики по кампанії;
- перегляд статистики по банеру;
- реєстрація сайту для отримання ролі вебмайстра.

Зареєстрований та авторизований у системі користувач, який має роль вебмайстра, має такі самі можливості, як і рекламодавець, а також додаткові можливості:

- перегляд статистики вебмайстра;
- реєстрація сайтів;
- створення рекламних блоків;
- редагування рекламних блоків;
- отримання коду блока;
- перегляд статистики рекламного блоку;

- запуск рекламного блоку;
- налаштування рекламного блоку.

3.2. Загальний опис системи

Після огляду аналогічних систем та проведення аналізу переваг та недоліків цих систем, було вирішено розробити веб-платформу для розміщення реклами, яка буде орієнтована на рекламу криптовалют, а також з оплатою послуг у криптовалюті, а саме Bitcoin.

Розроблена система є серверною частиною веб-платформи, яка має багато функціональних можливостей. Клієнтська частина має легкий та інтуїтивно зрозумілий інтерфейс який відповідає за взаємодію з сервером.

На головній сторінці веб-платформи користувачу пропонується увійти до системи, ввівши свої логін та пароль, або зареєструватися. Під час реєстрації користувач вказує електронну адресу та пароль, електронна адреса є логіном.

Зареєстровані в системі клієнти можуть мати два типу ролей: рекламодавець та вебмайстер. Після реєстрації користувача надається роль рекламодавця. Для отримання ролі вебмайстра потрібно зареєструвати свій сайт у системі. Також, користувач може мати дві ролі одночасно, як рекламодавця, так і вебмайстра, але це не змушує його і рекламувати і показувати рекламу на своєму сайті.

Роль рекламодавця надає користувачу можливість для створення рекламних кампаній. Існує два типи рекламних кампаній: CPC та CPM. Після цього у рекламній кампанії потрібно створити рекламний банер, обраного на вибір типу (текстовий, графічний або адаптивний). Після створення банера, лише потрібно поповнити баланс рекламної кампанії з особистого балансу користувача, та за бажання налаштувати таргетинг реклами. Тепер кампанія готова до запуску.

Роль вебмайстра надає можливість створювати рекламні блоки, які в свою чергу потрібно розміщувати на зареєстрованому у системі сайті. У

цьому блоці буде відображатися реклама рекламодавців. За кожне натискання по рекламі у рекламному блоці або за кожні 1000 переглядів, вебмайстер будет отримувати гроші, від рекламодавця, якому належала реклама.

Кожен користувач може перейти на сторінку особистого кабінету та заповнити дані про себе, а також перейти на сторінку балансу, та поповнити баланс, або ж вивести гроші. Якісна статистика дасть рекламодавцям можливість відслідковувати прогрес своїх кампаній та витрати на них, або заробіток для вебмайстрів.

3.3. Архітектура системи

Для розробки веб-платформи було вирішено використовувати концепцію клієнт-серверної архітектури.

Клієнт-серверна архітектура є одним із архітектурних стандартів програмного забезпечення та є основною концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними. Ця архітектура передбачає такі основні компоненти:

- сервери, що надають інформацію програмам, які звертаються до них;
- клієнти, програми, які звертаються до серверів за інформацією;
- мережа, як відповідає за взаємодію клієнта та сервера.

Сервери є незалежними, клієнти так само працюють паралельно і є незалежними один від одного. Типовою є ситуація, коли один сервер одночасно обробляє запити від багатьох різних клієнтів.

Архітектурна модель клієнт-серверної взаємодії визначається розподілом обов'язків між сервером і клієнтом. Можна виділити три рівні обов’язків:

- рівень відображення даних, який і є інтерфейсом користувача та відповідає за представлення даних користувачеві і введення від нього керуючих команд;

- прикладний рівень, який реалізує основну логіку і на якому здійснюється необхідна обробка інформації;
- рівень керування даними, який забезпечує зберігання та оновлення даних та доступ до них.

Зв'язок між рівнями здійснюється за певними правилами, які називаються протоколом взаємодії.

Рис. 3.1. Триланкова клієнт-серверна архітектура

Розробка серверної частини платформи, означає реалізацію прикладного рівня та рівня керування даними.

3.3.1. *Використання Node.js та фреймворку Express.js*

При розробці серверної частини веб-платформи було використано платформу node.js та фреймворк для побудови веб-додатків Express.js. Це найпростіший та найпопулярніший фреймворк для побудови веб-додатків та API, з відкритим кодом.

Express було використано для полегшення та пришвидшення розробки REST API веб-платформи для розміщення реклами. Express надає зрозумілі та зручні інструменти для розробки, має вичерпну документацію та велику підтримку від спільноти розробників.

Файл server.js є основним. З нього запускається сервер. В цьому файлі підключаються та налаштовуються різні модулі, які потрібні для роботи серверної частини веб-платформи, а точніше:

- налаштування підключення до бази даних MongoDB;

- налаштування сховища сесій;
- додавання модулів для парсингу запитів, що надходять на сервер;
- налаштування аутентифікації та авторизації користувачів за допомогою модуля passport.js;
- використання csrf токену для захисту;
- підключення усіх маршрутів API (файли routes.js та routes-auth.js).

Файл routes-auth.js відповідає за реєстрацію користувачів, вхід до системи, та вихід з системи. Логіка входу та реєстрації реалізується за допомогою стороннього модуля passport.js. Вона описується у допоміжному файлі passport.js.

Файл routes.js містить усі REST API, які задовольняють усім поставленим вище вимогам до функціональних можливостей системи.

3.3.2. *Схема бази даних*

У комплект до Node.js та Express.js була обрана система контролю базами даних MongoDB, яка дуже добре підходить до цього стеку технологій. Як графічний інтерфейс користувача для бази даних MongoDB було використано Robo 3T. Robo 3T – це безкоштовний інструмент для керування MongoDB з графічним інтерфейсом та вбудованою оболонкою, також програма доступнах для роботи на усіх платформах (Linux, MacOS, Windows).

ER-діаграма основних сутностей бази даних:

Рис. 3.2. Основні сутності бази даних

Основні сутності бази даних:

- users – містить інформацію про зареєстрованих користувачів, таку як зашифрований пароль, електронна адреса, статус підтвердження акаунта, статус активності користувача, статус чи є користувач вебмайстром, дата реєстрації, адреса для поповнення рахунку, повне ім’я, телефон, країну та додатковий електронний адрес;
- campaigns – містить інформацію про рекламну кампанію: унікальний ідентифікатор користувача, якому належить кампанія,

назва кампанії, дату створення кампанії, тип кампанії, а також великий список таргетингових налаштувань:

- мова;
- список країн в яких показувати рекламу;
- список країн в яких не показувати рекламу;
- ціна купівлі показів з інших країн, яких немає є у списку країн, в яких показувати рекламу;
- список платформ для показу реклами на телефонах;
- список платформ для показу реклами на комп'ютерах;
- через який час можна показувати рекламу повторно;
- в який час показувати рекламу;
- обмеження на кількість показів одному користувачу;
- список сайтів на яких показувати рекламу;
- список сайтів на яких не показувати рекламу;
- ціна купівлі показів з інших сайтів, яких немає у списку сайтів, на яких показувати рекламу.

o adverts – містить інформацію про рекламні банери користувачів, а саме унікальний ідентифікатор користувача, якому належить банер, унікальний ідентифікатор кампанії, якій належить банер, дату модифікації банера, інформацію про те чи є банер новим, схваленим, дату схвалення банера, тип банера, категорії, а також детальну інформацію про контент банера:

- мова;
- зображення (для графічних банерів);
- назва;
- два описи (для текстових банерів);
- посилання, яке показується на банері (для текстових банерів);
- посилання для переходу при натисканні на банер;
- ціна;

- заголовок (для адаптивних банерів);
- назву фірми (для адаптивних банерів);
- зображення логотипу фірми (для адаптивних банерів);
- зображення фону (для адаптивних банерів).

- publisher_sites – містить інформацію про сайти користувачів, таку як айді користувача, якому належить сайт, URL сайту, унікальний ідентифікатор сайту, категорії, до яких належить сайт, мову, опис, дату створення, дату схвалення, а також статуси, які показують чи схвалений сайт та чи активний;
- advertblocks – містить інформацію про рекламні блоки користувачів, а саме унікальний ідентифікатор користувача, якому належить блок, назву, стиль блоку, статус про активність блоку, тип блоку, дату модифікації блоку, мінімальну ціну реклами для цього блоку, а також деякі налаштування для реклами яка буде показуватися у цьому блоці:
  - дозвіл на показ текстових банерів;
  - дозвіл на показ банерів зображень;
  - заборонені категорії для показу;
  - заливку для банера при відсутності реклами.

3.4. Висновки

В даному розділі був проведений загальний опис розробленої серверної частини веб-платформи для розміщення реклами, було перелічено основні функціональні можливості для неавторизованих та авторизованих користувачів. Усі можливості були повністю реалізовані у серверній частині системі.

Були розглянуті використані технології, зокрема використання стеку технологій Node.js, Express.js, MongoDB, що є одним із найпопулярніших та найбільш ефективніших на сьогоднішній день для розробки веб-додатків та API.

Також були детально описані основні сутності бази даних та зв’язки між ним для веб-платформи.

4. АНАЛІЗ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості реалізації серверної частини за допомогою Express

Відповідно до обраного архітектурного шаблону клієнт-сервер, у дипломному проекті було реалізовано сервер, а також описана взаємодія сервера з базою даних та клієнтом.

Взаємодія сервера з базою даних MongoDB організована за рахунок підключення стороннього модуля, який називається mongoose. Для взаємодії потрібне описання моделей спроектованих сутностей бази даних, за допомогою mongoose схеми. На рис. 4.1 зображено приклад схеми користувача, що по своїй суті є JSON-об’єктом.

```
// define the schema for our user model
var userSchema = mongoose.Schema({
  full_name          : { type: String },
  contact_info       : {
    additional_email : { type: String },
    country          : { type: String },
    contact_phone    : { type: String },
  },
  local              : {
    email            : { type: String, index: true, unique: true },
    password         : String
  },
  status             : {
    active_publisher : {type: Boolean, index: true, default: false },
    active           : { type: Boolean, index: true, default: null },
    approved         : { type: Boolean, index: true, default: false },
    approval_date    : { type: Date}
  },
  lastLogin          : { type: Date },
  registered         : { type: Date, default: Date.now},
  advDate: { type: Date },
  referrer           : {
    id               : { type: String },
    url              : { type: String },
    limit            : { type: Number, default: 20 }
  },
  balance            : {
    btc_address      : { type: String, index: true }
  }
});
```

Рис. 4.1. Приклад схеми користувача

До схем можна додавати свої методи, які будуть описувати поведінку об’єкта (рис. 4.2).

```
// generating a hash
userSchema.methods.generateHash = function(password) {
  return bcrypt.hashSync(password, bcrypt.genSaltSync(8), null);
};
```

Рис. 4.2. Приклад метода схеми користувача

Таким самим чином, за допомогою схем у вигляді JSON об’єктів, описані усі сутності бази даних, з якими взаємодія серверна частина платформи, а саме:

- користувач;
- кампанія;
- рекламний банер;
- рекламний блок;
- сайт;
- та інші.

Після створення схеми, потрібно перетворити її на модель за допомогою функції model бібліотеки mongoose і дати їй назву. Модель є ніби класом, за допомогою якого ми будуємо документ. Кожен документ створений з цієї моделі, буде мати усі властивості та поведінку, прописані у схемі.

Бібліотека mongoose має реалізований увесь потрібний розробнику функціонал для пошуку, додавання, оновлення та видалення даних з бази даних. Бібліотека надає такі основні функції для використання:

- save() – збереження об’єкту;
- find() – пошук усіх об’єктів, які відповідають критеріям фільтрації;
- findOne() – пошук одного об’єкта, який відповідає критеріям фільтрації;
- findById() – пошук об’єкта по значенню поля _id;
- remove() – видаляє усі об’єкти, які відповідають критеріям фільтрації;

- findOneAndDelete() – видаляє один об’єкт, який відповідає критеріям фільтрації;
- findByIdAndDelete() – видаляє об’єкт по значенню поля _id;
- updateOne() – оновлює один документ, який відповідає критеріям фільтрації;
- updateMany() – оновлює усі документ, які відповідають критеріям фільтрації;
- findByIdAndUpdate() – оновлює об’єкт по значенню поля _id.

Можна помітити, що назви функцій повністю описують операції які вони виконуються. За допомогою перелічених функцій бібліотеки mongoose, сервер проводить усі потрібні операції над даними у базі.

З клієнтською частиною сервер спілкується по створенному програмному інтерфейсу.

Приведена нижче діаграма (рис. 4.3) показує основний потік даних і елементи, які необхідно реалізувати під час обробки HTTP-запитів/відповідей. Окрім представлих там маршрутів на діаграмі показані контролери. Контролери – це функції які відділяють код для маршрутизації запитів від кода, який обробляє запити.

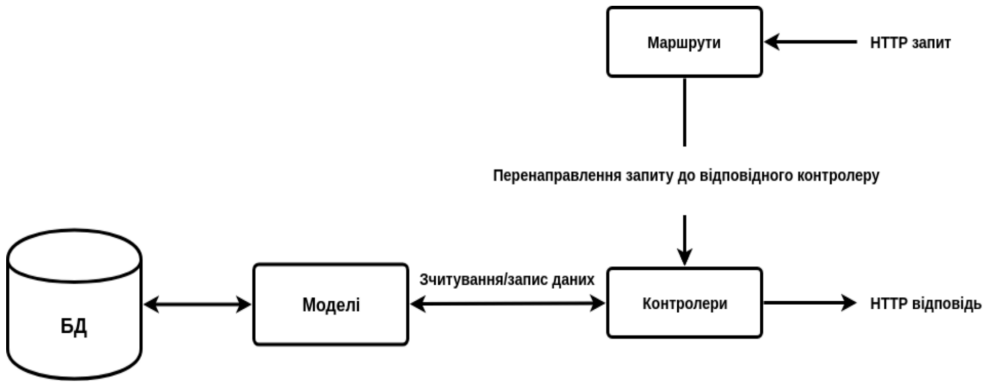


Рис. 4.3. Основний потік даних та елементи необхідні для обробки HTTP-запитів/відповідей

Після моделей розробляються такі елементи:

- маршрути для перенаправлення запитів відповідним функціям-контролерам;
- контролери-функції для отримання потрібних даних з моделей, та обробки даних.

Маршрути – це частина коду Express, яка пов’язує HTTP запити, URL шляхи з функцією, що обробляє цей шлях.

HTTP запити поділяються на:

- GET;
- POST;
- DELETE;
- PUT.

В основному було використано лише два типи запитів: GET та POST.

Контролери відповідають за обробку запитів користувачів та передачу відповіді на запит. Були розроблені такі основні контролери:

- registrationController – контролер, який відповідає за регістрацію користувачів у системі;
- authController – контролер, який відповідає за авторизацію користувачів у системі;
- userController – контролер користувача, відповідає за:
  - отримання даних профіля користувача;
  - збереження інформації користувача.
- campaignController – контролер кампанії, відповідає за:
  - створення кампаній;
  - отримання інформації по кампаніям користувача;
  - отримання статистики по кампаніям;
  - редагування налаштувань таргетингу кампаній;
  - запуск та зупинку кампаній;
  - поповнення кампаній;
  - редагування назви кампанії.

- advertController – контролер рекламних банерів, відповідає за:
  - створення рекламних банерів;
  - редагування рекламних банерів;
  - отримання інформації про рекламні банери користувача;
  - отримання статистики по рекламним банерам;
  - запуск рекламних банерів.
- siteController – контролер сайтів, відповідає за:
  - регістрацією сайтів у системі;
  - отриманню інформації про сайти зареєстровані у системі.
- adblockController – контролер рекламних блоків, відповідає за:
  - створення рекламних блоків;
  - редагування рекламних блоків;
  - отримання статистики по рекламним блокам;
  - налаштування рекламних блоків;
  - отримання інформації про рекламні блоки.
- інші контролери, які відповідають за допоміжні функції не пов'язані з вище перерахованими контролерами.

Приклад вигляду частини контролеру кампанії рис. 4.4.

```
'use strict';

const express        = require('express');
const controller     = require('./campaign.controller');
const helpers        = require('./../../components/controller-helpers');

let router           = express.Router();

router.get('/gettoblacklist/:campaignId',                helpers.isLoggedIn, helpers.userHasCampaign, controller.getBlackListByCampId);
router.post('/addtoblacklist/:campaignId',               helpers.isLoggedIn, helpers.userHasCampaign, controller.addToBlackListByCampId);
router.post('/addtowhitelist/:campaignId',               helpers.isLoggedIn, helpers.userHasCampaign, controller.addToWhiteListByCampId);
router.post('/addtoblacklist/bulk/:campaignId',          helpers.isLoggedIn, helpers.userHasCampaign, controller.addToBlackListBulkByCampId);
router.post('/addtowhitelist/bulk/:campaignId',          helpers.isLoggedIn, helpers.userHasCampaign, controller.addToWhiteListBulkByCampId);
router.post('/deletefromblacklist/selected/:campaignId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromBlackListSelectedByCampId);
router.post('/deletefromwhitelist/selected/:campaignId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromWhiteListSelectedByCampId);
router.post('/eneble/:campaignId',                       helpers.isLoggedIn, helpers.userHasCampaign, controller.enebleBlackListByCampId);
router.post('/traffic/status/:campaignId',               helpers.isLoggedIn, helpers.userHasCampaign, controller.setTrafficStatusByCampId);
router.post('/traffic/percent/:campaignId',              helpers.isLoggedIn, helpers.userHasCampaign, controller.setTrafficPercentByCampId);
router.delete('/deletefromblacklist/:campaignId/:sourceId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromBlackList);
router.delete('/deletefromwhitelist/:campaignId/:sourceId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromWhiteList);

module.exports = router;
```

Рис. 4.4. Вигляд частини контролеру кампанії

4.2. Тестування веб-платформи

Тестування програмного забезпечення є однією з частин створення програмного продукту та є однією з основних стадій життєвого циклу програмного забезпечення. Тестування програмного забезпечення – це процес, під час якого перевіряється відповідність заявлених вимог до продукту і реально реалізованих функціональних можливостей. Здійснюється шляхом спостереження за роботою продукту в штучно створених ситуаціях і на обмеженому наборі тестів, обраних певним чином.

Перш за все тестування спрямоване на пошук помилок, перевірку коректності роботи програми та перевірку зручності графічного інтерфейсу користувача.

Під час розробки програмного забезпечення, передбаченого дипломним проектом, тестування виконувалося після кожного етапу написання коду певних функціональних частин системи.

Одним із способів тестування було ручне тестування. За допомогою цього способу, в поля для вводу даних вводилися коректні або некоректні дані для перевірки роботи програми у різних ситуаціях. Під час вводу некоректних даних була виявлена необхідність підказок для користувача, щоб інтерфейс був більш зрозуміліший.

За допомогою браузера Chrome та вбудованих інструментів для розробників, робилась перевірка правильності запитів, що надходять до сервера, та перевірка відповіді сервера.

За допомогою графічного інтерфейсу Robo 3T для бази даних MongoDB, перевірялось коректне записування даних до бази, а також оновлення даних у базі та видалення даних.

За допомогою тестування, була отримана важлива інформація, яка допомогла знайти та виправити проблемні місця в логіці роботи серверної частини веб-платформи, а також інформація, щодо майбутнього вдосконалення програмного забезпечення веб-платформи.

4.3. Порівняння розробки з існуючими аналогами

В першому розділі дипломного проекту було розглянуто декілька різноманітних аналогів сучасних сервісів для розміщення реклами в мережі Інтернет.

Всі аналоги були порівнянні між собою, а саме були виявлені їх переваги та недоліки.

Розроблена веб-платформа відповідає усім визначеним вимогам, які були поставлені на основі аналізу переваг аналогів. А саме:

- зручність та зрозумілість користувацького інтерфейсу;
- гнучкість налаштування таргетингу реклами;
- детальна статистика по рекламним кампаніям користувача;
- детальна статистика по рекламним блокам користувача.

Також, під час розробки веб-платформи були усунуті деякі недоліки аналогічних систем:

- обмеження на рекламу криптовалют, а також сайтів та сервісів пов’язаних з ними;
- високий поріг входу для вебмайстрів, тобто не важливо як багато користувачів відвідує сайт вебмайстра;
- відсутня сума мінімального депозиту, поповнити акаунт можливо на будь-яку суму.

Проаналізувавши розроблене програмне забезпечення, можна впевнено стверджувати, що воно являється достойним конкурентом для вже існуючих аналогів платформ для розміщення реклами.

4.4. Рекомендації щодо подальшого вдосконалення

Розроблене програмне забезпечення для серверної частини веб-платформи для розміщення реклами може використовуватися рекламними агенціями та компаніями, так як воно містить багато функціональних можливостей для повноцінної роботи. Проте, так як бажання користувачів

швидко зростають, цього може не вистачити. Тому потрібне постійне вдосконалення програмного забезпечення.

Отже веб-платформу можна вдосконалити додаванням нових можливостей, що будуть мати попит як у рекламодавців так і у вебмайстрів:

- додавання нових типів реклами, наприклад можливість створити відео рекламу;
- додавання нових розмірів рекламних блоків;
- постійне покращення користувацького інтерфейсу, залежно від відгуків користувачів системи;
- покращення кодової бази серверної частини;
- оптимізація запитів до сервера;
- оптимізація запитів до бази даних;
- додавання підтвердження електронної адреси користувача;
- додавання двофакторної аутентифікації, для покращення захисту акаунтів користувачів;
- додавання ботів, для відповіді на прості питання користувачів;
- додавання системи сповіщень користувача, про закінчення коштів на рекламній кампанії;
- додавання генерації щотижневих звітів, що будуть автоматично надходити до користувача.

ВИСНОВКИ

Метою цього дипломного проекту була розробка програмного забезпечення серверної частини веб-платформи для розміщення реклами.

Після огляду та аналізу існуючих аналогів систем для розміщення реклами, було зроблено висновок, що доцільно розробляти систему у вигляді веб-платформи, що матиме певний набір функцій та зручний і інтуїтивно зрозумілий інтерфейс користувача.

На основі проведеного аналізу різноманітних засобів реалізації, було вирішено створити серверну частини веб-платформи за допомогою мови JavaScript, а саме платформи Node.js та фреймворку Express.js. Також була використана СКБД MongoDB, яка гармонійно доповнює цей стек технології. Також була використана концепція клієнт-серверної архітектури.

Розроблена веб-платформа:

- має зручний, інтуїтивно зрозумілий користувацький інтерфейс;
- надає можливість реєстрації користувача, та отримувати їм два типи ролей, а саме рекламодавець та вебмайстер;
- надає можливість авторизованим користувачам легко та швидко створювати рекламні кампанії, рекламні банери та рекламні блоки;
- надає можливість авторизованим користувачам гнучко налаштовувати таргетинг рекламних кампаній, а також налаштовувати рекламні блоки;
- надає авторизованим користувачам інструменти для детального перегляду статистичної інформації пов’язаної з рекламною кампанією, рекламним банером або рекламним блоком.

Розробка програмного забезпечення виконана у повному обсязі та повністю відповідає поставленим вимогам до нього.

Тестування веб-платформи виконано у відповідності до затвердженої програми та методики тестування .

Використання створеної веб-платформи створить для рекламодавців та вебмайстрів велику мережу для розміщення якісної реклами на великій кількості сайтів. Користувачам не потрібно мати спеціальних знань чи багато часу для початку роботи з веб-платформою.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Електронний маркетинг [Електронний ресурс] — дата візиту 10.12.2018 — Режим доступу до ресурсу: https://bit.ly/2AJEZ66
2. Інтернет-маркетинг [Електронний ресурс] — дата візиту 10.12.2018 — Режим доступу до ресурсу: https://bit.ly/2WVbf2o
3. Google Ads [Електронний ресурс] — дата візиту 22.12.2018 — Режим доступу до ресурсу: https://bit.ly/2RZiT5j
4. Google Ads [Електронний ресурс] — дата візиту 22.12.2018 — Режим доступу до ресурсу: https://bit.ly/2XvEfuE
5. Яндекс.Дірект [Електронний ресурс] — дата візиту 22.12.2018 — Режим доступу до ресурсу: https://bit.ly/2Wvg7Hi
6. AdBean [Електронний ресурс] — дата візиту 26.12.2018 — Режим доступу до ресурсу: https://bit.ly/2WnCwpP
7. AdBean огляд [Електронний ресурс] — дата візиту 26.12.2018 — Режим доступу до ресурсу: https://bit.ly/2I4WSQJ
8. Coinzilla [Електронний ресурс] — дата візиту 26.12.2018 — Режим доступу до ресурсу: https://bit.ly/2K63m40
9. Coinzilla огляд [Електронний ресурс] — дата візиту 26.12.2018 — Режим доступу до ресурсу: https://bit.ly/2IykFrr
10. Мобільний застосунок [Електронний ресурс] — дата візиту 15.01.2018 — Режим доступу до ресурсу: https://bit.ly/31knMM5
11. Веб-застосунок [Електронний ресурс] — дата візиту 15.01.2018 — Режим доступу до ресурсу: https://bit.ly/31mwo4X
12. Криптовалюта — що це? Бізнес в Інтернеті та електронна комерція [Електронний ресурс] — дата візиту 15.01.2018 — Режим доступу до ресурсу: https://bit.ly/2Xw5Yey
13. Інтернет-реклама як основний чинник просування товару в сучасних умовах [Електронний ресурс] — дата візиту 15.01.2018 — Режим доступу до ресурсу: https://bit.ly/2WxqcYY

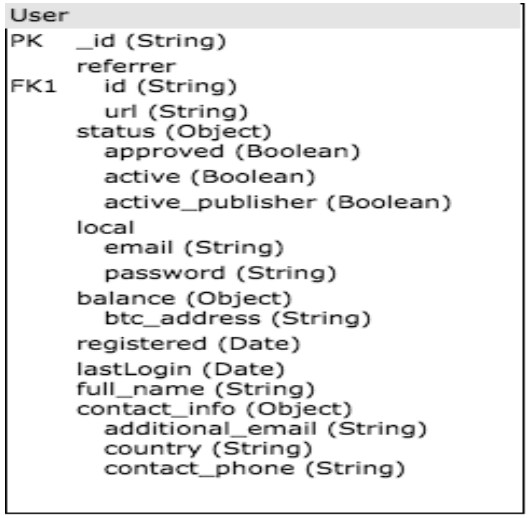
14. Інтернет-реклама як маркетинговий інструмент [Електронний ресурс] — дата візиту 28.01.2018 — Режим доступу до ресурсу: https://bit.ly/2MLEv7A
15. Інтернет-реклама та суспільство [Електронний ресурс] — дата візиту 28.01.2018 — Режим доступу до ресурсу: https://bit.ly/2Zltfk1
16. MDN JavaScript [Електронний ресурс] — дата візиту 13.02.2018 — Режим доступу до ресурсу: https://mzl.la/2Dj6juU
17. JavaScript wiki [Електронний ресурс] — дата візиту 13.02.2018 — Режим доступу до ресурсу: https://bit.ly/1RgM3Gw
18. EcmaScript [Електронний ресурс] — дата візиту 13.02.2018 — Режим доступу до ресурсу: https://bit.ly/2IAbM0e
19. Node.js [Електронний ресурс] — дата візиту 15.02.2018 — Режим доступу до ресурсу: https://bit.ly/1Yn5aol
20. Про Node.js [Електронний ресурс] — дата візиту 15.02.2018 — Режим доступу до ресурсу: https://bit.ly/2Zfo1Gg
21. Express.js [Електронний ресурс] — дата візиту 15.02.2018 — Режим доступу до ресурсу: https://bit.ly/2wMZc8k
22. Express.js wiki [Електронний ресурс] — дата візиту 15.02.2018 — Режим доступу до ресурсу: https://bit.ly/2QYbzHZ
23. PHP wiki [Електронний ресурс] — дата візиту 22.02.2018 — Режим доступу до ресурсу: https://bit.ly/1NjS96m
24. Можливості PHP [Електронний ресурс] — дата візиту 22.02.2018 — Режим доступу до ресурсу: https://bit.ly/2MAlchz
25. PHP переваги та недоліки [Електронний ресурс] — дата візиту 22.02.2018 — Режим доступу до ресурсу: https://bit.ly/2XzJhX1
26. Python [Електронний ресурс] — дата візиту 02.03.2018 — Режим доступу до ресурсу: https://bit.ly/1oDM6iq
27. Python wiki [Електронний ресурс] — дата візиту 02.03.2018 — Режим доступу до ресурсу: https://bit.ly/1Ty799n

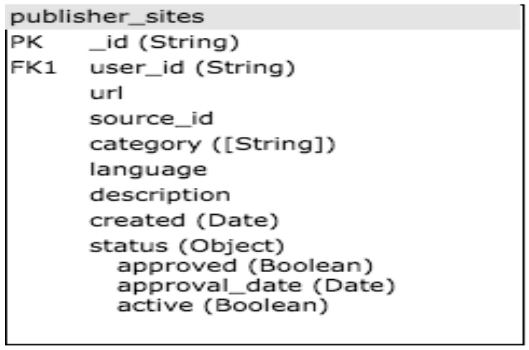
28. Python порівняно з PHP [Електронний ресурс] — дата візиту 02.03.2018 — Режим доступу до ресурсу: https://bit.ly/2I3MloN
29. Вибір кращого Nodes.js фреймворку [Електронний ресурс] — дата візиту 10.03.2018 — Режим доступу до ресурсу: https://bit.ly/2WtHa5r
30. Sails.js [Електронний ресурс] — дата візиту 10.03.2018 — Режим доступу до ресурсу: https://bit.ly/2WqXfZE
31. СКБД wiki [Електронний ресурс] — дата візиту 23.03.2018 — Режим доступу до ресурсу: https://bit.ly/2QFmar0
32. СКБД опис [Електронний ресурс] — дата візиту 23.03.2018 — Режим доступу до ресурсу: https://bit.ly/2wL38qm
33. Bitcoin [Електронний ресурс] — дата візиту 26.03.2018 — Режим доступу до ресурсу: https://bit.ly/2K2532k
34. MySQL [Електронний ресурс] — дата візиту 12.04.2018 — Режим доступу до ресурсу: https://bit.ly/1ndQR7k
35. MySQL wiki [Електронний ресурс] — дата візиту 12.04.2018 — Режим доступу до ресурсу: https://bit.ly/2JwDM8B
36. Основні переваги СУБД MySQL [Електронний ресурс] — дата візиту 12.04.2018 — Режим доступу до ресурсу: https://bit.ly/2WYOURp
37. MongoDB [Електронний ресурс] — дата візиту 19.04.2018 — Режим доступу до ресурсу: https://bit.ly/1NL0pyD
38. MongoDB wiki [Електронний ресурс] — дата візиту 19.04.2018 — Режим доступу до ресурсу: https://bit.ly/1PZ4ALV
39. MongoDB переваги [Електронний ресурс] — дата візиту 21.04.2018 — Режим доступу до ресурсу: https://bit.ly/2R4iYG0
40. MongoDB та MySQL - Коли що використовувати [Електронний ресурс] — дата візиту 21.04.2018 — Режим доступу до ресурсу: https://bit.ly/2wI6k6d
41. REST [Електронний ресурс] — дата візиту 05.05.2018 — Режим доступу до ресурсу: https://bit.ly/2WnvuS0

42. ER-діаграма [Електронний ресурс] — дата візиту 06.05.2018 — Режим доступу до ресурсу: https://bit.ly/2XBK9u2
43. Клієнт-серверна архітектура [Електронний ресурс] — дата візиту 09.05.2018 — Режим доступу до ресурсу: https://bit.ly/2wKDkum
44. Клієнт-серверна архітектура wiki [Електронний ресурс] — дата візиту 09.05.2018 — Режим доступу до ресурсу: https://bit.ly/2F1rOPR

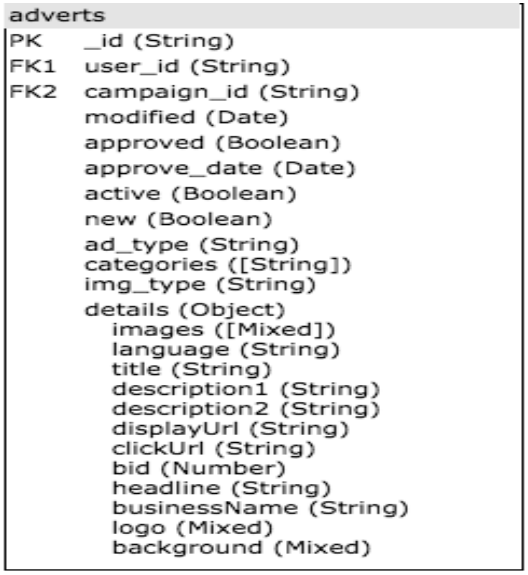
ДОДАТКИ

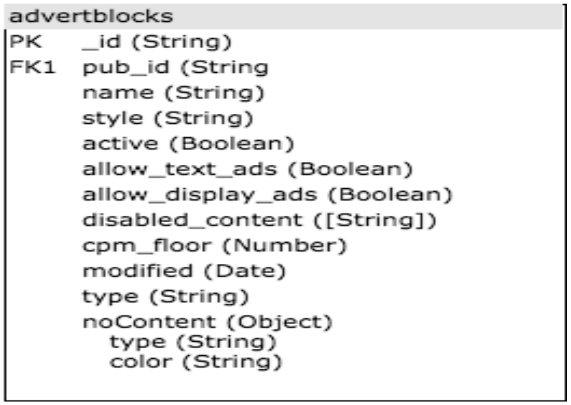
Додаток 1
Копії графічних матеріалів


User
PK _id (String)
referrer
FK1 id (String)
url (String)
status (Object)
approved (Boolean)
active (Boolean)
active_publisher (Boolean)
local
email (String)
password (String)
balance (Object)
btc_address (String)
registered (Date)
lastLogin (Date)
full_name (String)
contact_info (Object)
additional_email (String)
country (String)
contact_phone (String)

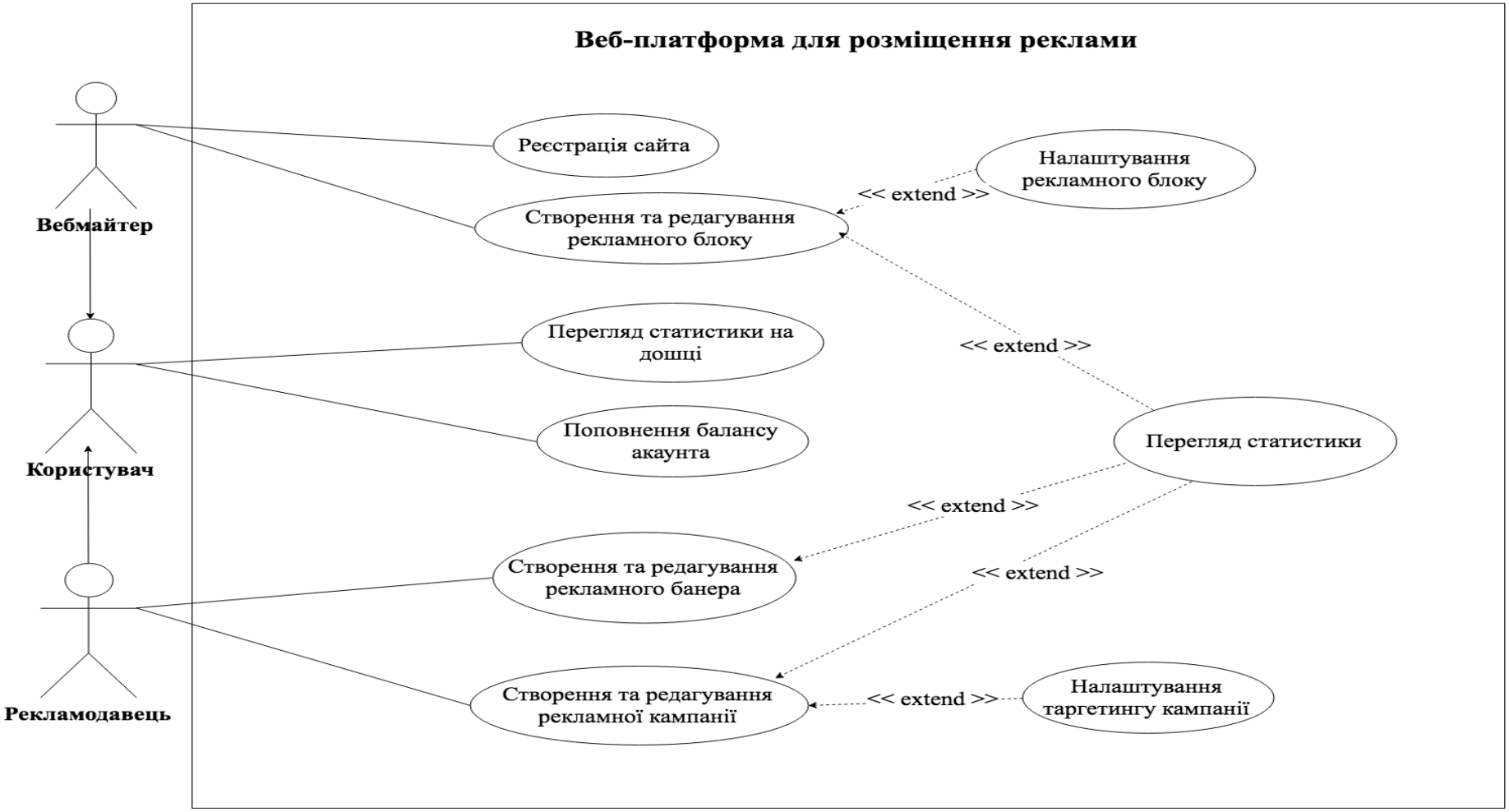

publisher_sites
PK _id (String)
FK1 user_id (String)
url
source_id
category ([String])
language
description
created (Date)
status (Object)
approved (Boolean)
approval_date (Date)
active (Boolean)


campaigns
PK _id (String)
FK1 user_id (String)
campaign_name (String)
created (Date)
type (String)
finance.daily_limit (String)
targeting_options (Object)
language (String)
locations_preferred (Boolean)
locations_preferred_list ([String])
locations_excluded (Boolean)
locations_excluded_list ([String])
buy_other (Boolean)
buy_other_value (Number)
display_desktop (Boolean)
display_mobile (Boolean)
display_mobile_os (Object)
android (Boolean)
ios (Boolean)
winmobile (Boolean)
others (Boolean)
display_desktop_os (Object)
windows (Boolean)
linux (Boolean)
macos (Boolean)
others (Boolean)
ad_rerun (number)
display_time_range (Object)
from (Number)
to (Number)
frequency_capping (Boolean)
frequency_value (Number)
frequency_time (Number)
frequency_target (String)
blocked_sources ([String])
white_list ([String])
buy_other_forlist (Boolean)
buy_other_value_forlist (Number)
blacklist_enable (Boolean)

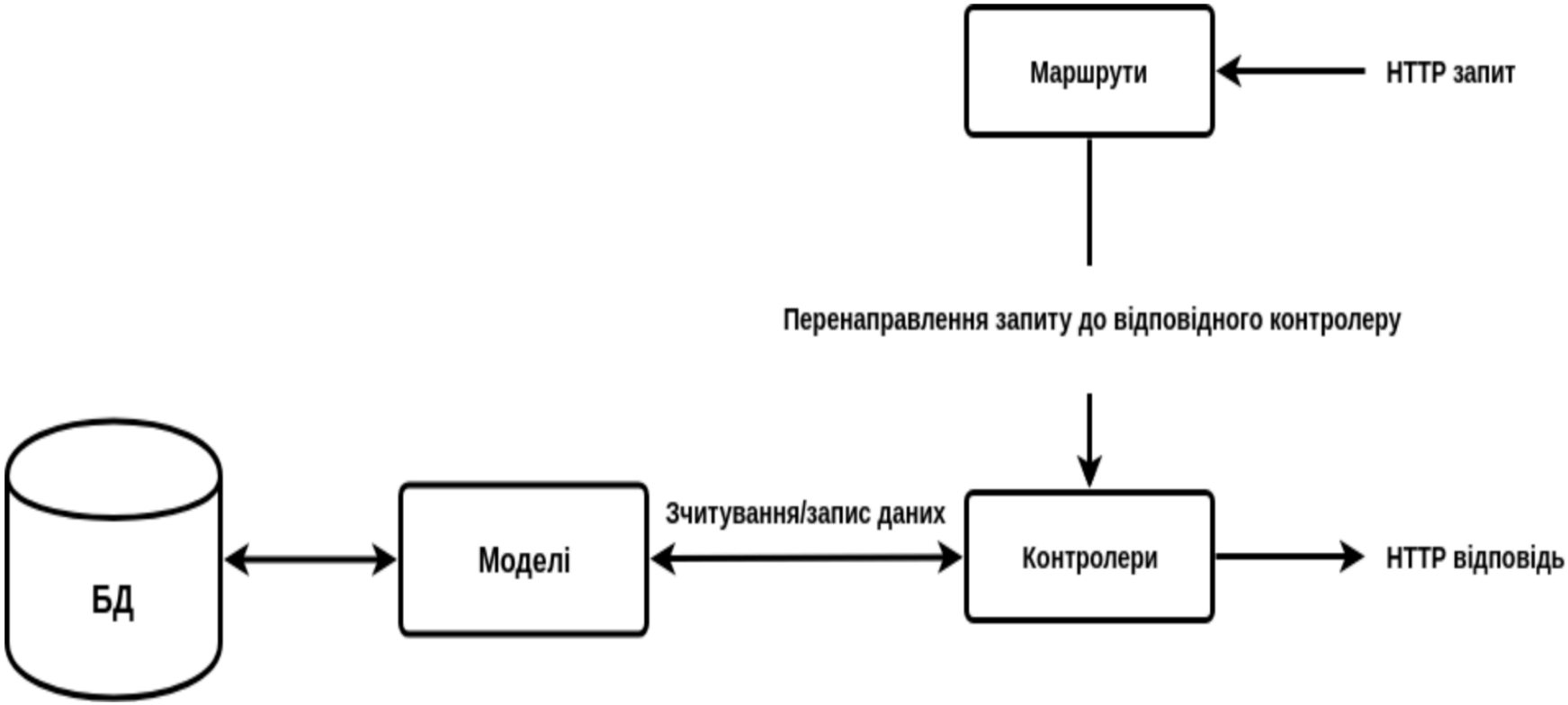

adverts
PK _id (String)
FK1 user_id (String)
FK2 campaign_id (String)
modified (Date)
approved (Boolean)
approve_date (Date)
active (Boolean)
new (Boolean)
ad_type (String)
categories ([String])
img_type (String)
details (Object)
images ([Mixed])
language (String)
title (String)
description1 (String)
description2 (String)
displayUrl (String)
clickUrl (String)
bid (Number)
headline (String)
businessName (String)
logo (Mixed)
background (Mixed)


advertblocks
PK _id (String)
FK1 pub_id (String
name (String)
style (String)
active (Boolean)
allow_text_ads (Boolean)
allow_display_ads (Boolean)
disabled_content ([String])
cpm_floor (Number)
modified (Date)
type (String)
noContent (Object)
type (String)
color (String)

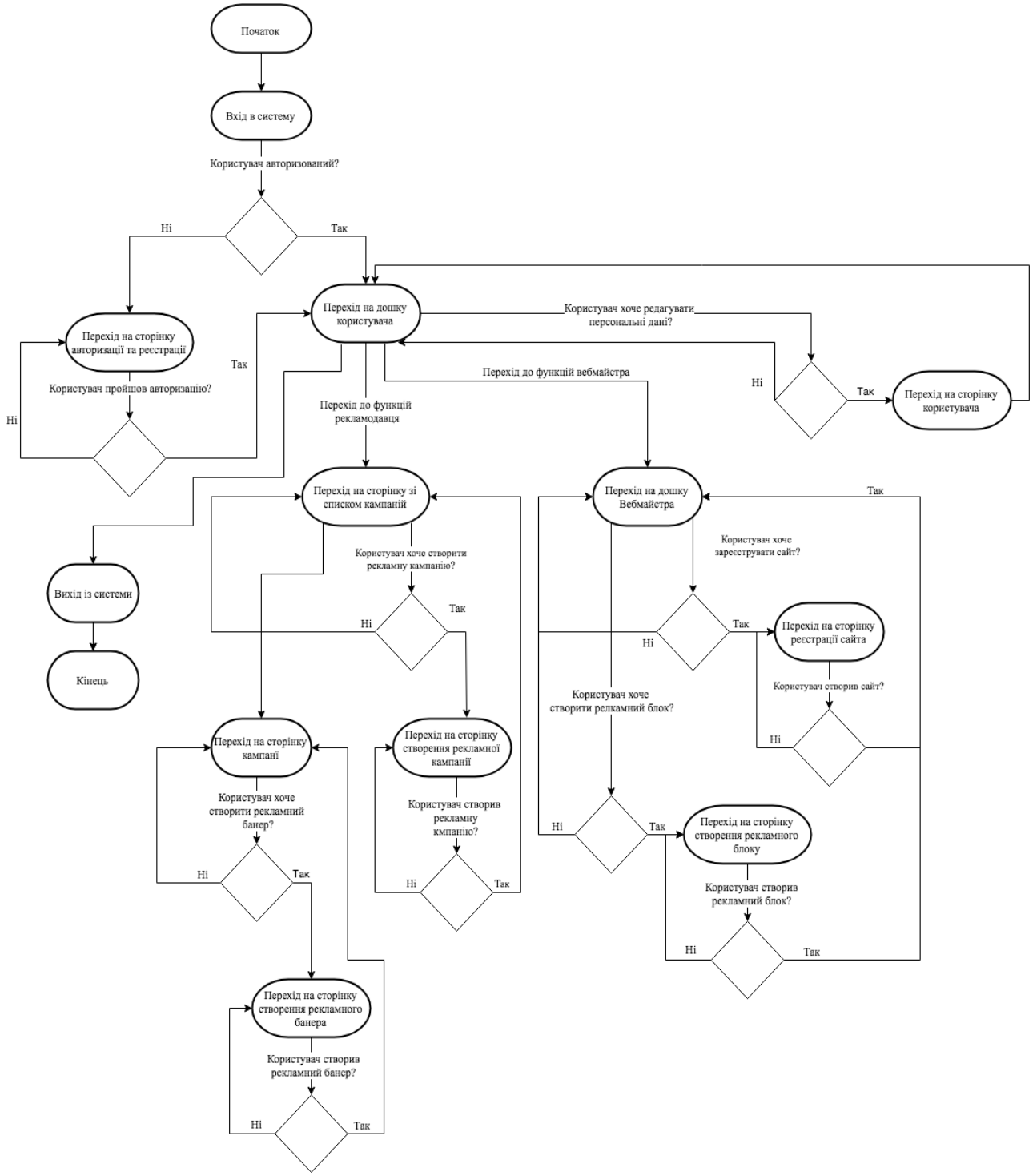
Веб-платформа для розміщення реклами. Серверна частина. Схема бази даних



Веб-платформа для розміщення реклами. Серверна частина. Діаграма варіантів використання системи



Дяченко Д.О., група КП-52



Дяченко Д.О., група КП-52

Додаток 2
Копія презентації

ВЕБ-ПЛАТФОРМА ДЛЯ РОЗМІЩЕННЯ РЕКЛАМИ. СЕРВЕРНА ЧАСТИНА

Виконав: студент групи КП-52 Дяченко Д.О.

Керівник: ст. викладач кафедри ПЗКС, к.т.н. Рибачок Н.А.

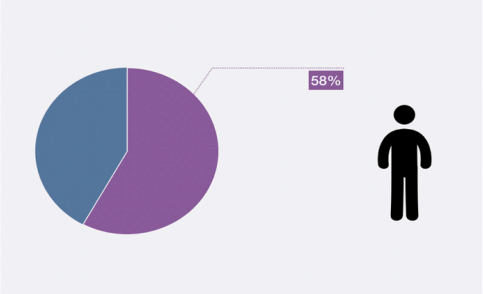
Київ – 2019

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробка програмного забезпечення серверної частини веб-платформи для розміщення реклами.

АКТУАЛЬНІСТЬ

За допомогою Інтернет-реклами освоюються нові ринки збуту. Реклама забезпечує можливість збільшення об'ємів продажів. Вона є дуже важливим і незамінним інструментом маркетингу, що встановлює, підтримує і розвиває комунікації між підприємством і споживачами. Реклама формує попит і керує ним.



АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ

Google Ads

Яндекс Директ

AdBean

coinzilla

АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ

	Гнучкість налаштувань	Спеціальна підготовка користувачів	Немає обмежень на рекламу криптовалют	Вузька спеціалізація платформи
Google Ads	+	+	–	–
Яндекс.Діректт	+	+	–	–
AdBean	–	–	+	+
Coinzilla	–	–	+	+

ФУНКЦІОНАЛЬНІ ВИМОГИ

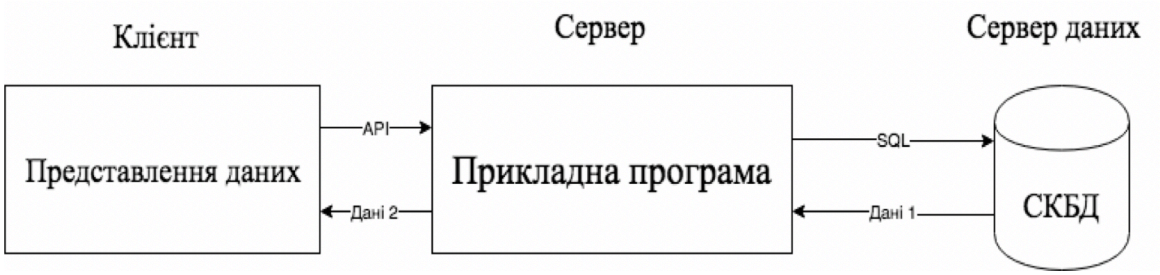
- Реєстрація та авторизація користувачів
- Реєстрація сайтів для вебмайстрів
- Створення рекламних кампаній для рекламодавців
- Створення рекламних банерів для рекламодавців
- Створення рекламних блоків для вебмайстрів
- Перегляд детальної статистики по кампаніях, банерах, блоках
- Гнучке налаштування рекламних банерів, кампаній та блоків

НЕФУНКЦІОНАЛЬНІ ВИМОГИ

- Підтримка роботи платформи у найбільш поширених веб-браузерах (Chrome, Opera, Safari, Firefox) та у мобільних браузерах також
- Зручність, зрозумілість та простота інтерфейсу користувача
- Наявність англійської локалізації

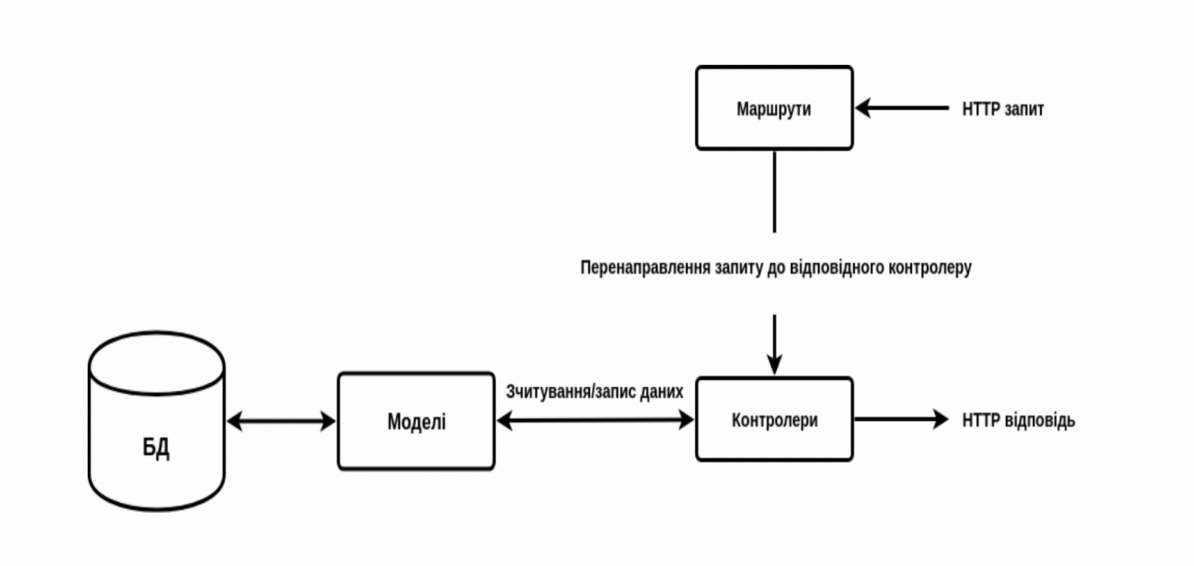
СТЕК ТЕХНОЛОГІЙ

АРХІТЕКТУРА СИСТЕМИ



Клієнт-серверна архітектура

СХЕМА ПОТОКУ ДАНИХ У СИСТЕМІ



ПРИКЛАД СТВОРЕННЯ МОДЕЛІ

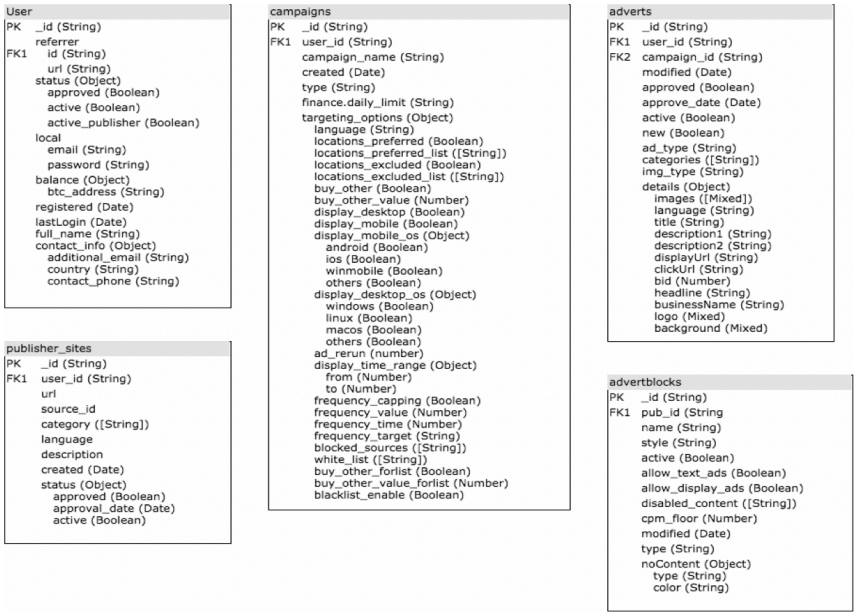
```
// define the schema for our user model
var userSchema = mongoose.Schema({
    full_name           : { type: String },
    contact_info        : {
      additional_email : { type: String },
      country          : { type: String },
      contact_phone    : { type: String },
    },
    local               : {
      email            : { type: String, index: true, unique: true },
      password         : String
    },
    status              : {
      active_publisher : {type: Boolean, index: true, default: false },
      active           : { type: Boolean, index: true, default: null },
      approved         : { type: Boolean, index: true, default: false },
      approval_date    : { type: Date}
    },
    lastLogin           : { type: Date },
    registered          : { type: Date, default: Date.now},
    advDate: { type: Date },
    referrer            : {
      id               : { type: String },
      url              : { type: String },
      limit            : { type: Number, default: 20 }
    },
    balance             : {
      btc_address      : { type: String, index: true }
    }
});
```

```
// generating a hash
userSchema.methods.generateHash = function(password) {
  return bcrypt.hashSync(password, bcrypt.genSaltSync(8), null);
};
```


ПРИКЛАД СТВОРЕННЯ МАРШРУТІВ

```
'use strict';

const express        = require('express');
const controller     = require('./campaign.controller');
const helpers        = require('./../../components/controller-helpers');

let router           = express.Router();

router.get('/gettoblacklist/:campaignId',                helpers.isLoggedIn, helpers.userHasCampaign, controller.getBlackListByCampId);
router.post('/addtoblacklist/:campaignId',               helpers.isLoggedIn, helpers.userHasCampaign, controller.addToBlackListByCampId);
router.post('/addtowhitelist/:campaignId',               helpers.isLoggedIn, helpers.userHasCampaign, controller.addToWhiteListByCampId);
router.post('/addtoblacklist/bulk/:campaignId',          helpers.isLoggedIn, helpers.userHasCampaign, controller.addToBlackListBulkByCampId);
router.post('/addtowhitelist/bulk/:campaignId',          helpers.isLoggedIn, helpers.userHasCampaign, controller.addToWhiteListBulkByCampId);
router.post('/deletefromblacklist/selected/:campaignId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromBlackListSelectedByCampId);
router.post('/deletefromwhitelist/selected/:campaignId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromWhiteListSelectedByCampId);
router.post('/eneble/:campaignId',                       helpers.isLoggedIn, helpers.userHasCampaign, controller.enebleBlackListByCampId);
router.post('/traffic/status/:campaignId',               helpers.isLoggedIn, helpers.userHasCampaign, controller.setTrafficStatusByCampId);
router.post('/traffic/percent/:campaignId',              helpers.isLoggedIn, helpers.userHasCampaign, controller.setTrafficPercentByCampId);
router.delete('/deletefromblacklist/:campaignId/:sourceId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromBlackList);
router.delete('/deletefromwhitelist/:campaignId/:sourceId', helpers.isLoggedIn, helpers.userHasCampaign, controller.deleteFromWhiteList);

module.exports = router;
```

СХЕМА БАЗИ ДАНИХ



ВАРІАНТИ ВИКОРИСТАННЯ СИСТЕМИ

ТЕСТУВАННЯ ВЕБ-ПЛАТФОРМИ

- Динамічне ручне тестування на введення коректних, граничних або недопустимих значень даних, у полі, що дозволено редагувати
- Динамічне ручне тестування на відповідність функціональним вимогам
- Тестування стабільності роботи у різних умовах
- Тестування швидкості відповіді на запити
- Тестування програмного інтерфейсу платформи

ПОРІВНЯННЯ З ІСНУЮЧИМИ АНАЛОГАМИ

Розроблена веб-платформа відповідає визначеним вимогам, які були поставлені на основі аналізу переваг аналогів, а саме:

- Зручність та зрозумілість користувацького інтерфейсу
- Гнучкість налаштувань таргетингу системи
- Детальна статистика по рекламним кампанія, банерам та блокам

Також були усунуті деякі недоліки аналогічних систем:

- Обмеження на рекламу криптовалют
- Високий поріг входу для вебмайстрів
- Відсутній мінімальний депозит

РЕКОМЕНДАЦІЇ ЩОДО ПОДАЛЬШОГО ВДОСКОНАЛЕННЯ

- Додавання нових типів реклами
- Додавання нових розмірів рекламних блоків
- Додавання адаптивних рекламних блоків
- Оптимізація запитів до бази даних
- Додавання підтвердження електронної адреси
- Додавання двофакторної аутентифікації, для підвищення захищеності акаунтів користувачів
- Додавання системи генерації щотижневих звітів
- Додавання ботів, для відповідей на прості запитання користувачів

ВИСНОВКИ

ВИСНОВКИ

Розроблено програмне забезпечення серверної частини веб-платформи для розміщення реклами.

Під час роботи над дипломним проектом було виконано:

- Аналіз аналогів
- Збір вимог
- Аналіз та вибір засобів реалізації
- Розробку архітектури, структури БД
- Тестування
- Створення супровідної документації
- Формулювання рекомендацій

ПЕРЕВІРКА НА УНІКАЛЬНІСТЬ

РОЗДІЛ	УНІКАЛЬНІСТЬ
ВСТУП	72%
Розділ 1	83%
Розділ 2	76%
Розділ 3	93%
Розділ 4	98%
ВИСНОВКИ	97%
Диплом повністю	82%

Дякую за увагу!

Додаток 3
Лістинг програми

```
   app.get('/login',             function(req,             res)             {
if                                (req.user)                                  {
  res.redirect('/');
}                                    else                                    {
  res.render('login.ejs',                                                    {
    message:                                          req.flash('loginMessage'),
    user:                                                                 req.user
  });
}
});

app.post('/login', function(req, res, next) {
  if (typeof req.body.email === 'undefined' || req.body.email.length == 0){
     return res.render('404.ejs', {msg:'Email and password are require
fields'});
  }

  passport.authenticate('local-login', function(err, user, info) {
    if(err || (!user && info)) {
      if (!info) {
        return res.render('404.ejs', {msg:err.message});
      }
      if (info === 'NOT_CORRECT_PASSWORD') {
        return res.render('404.ejs', {msg:'Incorrect password'});
      }
      if (info === 'NO_USER_FOUND') {
        return res.render('404.ejs', {msg:'User not found'});
      }
    }

    req.login(user, function(err) {
      if (err) {
        return next(err);
      }
      return res.redirect('/dashboard');
    });
  })(req,res,next);

});

app.get('/signup', function(req, res) {
  if(req.user){
    res.redirect('/');
} else {
  res.render('signup.ejs', {
    message: req.flash('signupMessage'),
    user: req.user
  });
}
});

app.get('/signupdone', function(req, res) {
  if(req.user){
    res.redirect('/');
  } else {
    res.render('signupdone.ejs', {user: null});
  }
});

app.post('/signup', function(req, res, next) {
    if (typeof req.body.email === 'undefined' || req.body.email.length == 0){
     return res.status(500).send('Email and password are require fields');
```

```
    }
  passport.authenticate('local-signup', function(err, user, info) {
    if (err) {
      res.status(500).send(err);
    } else {
      res.status(200).send({message: 'You are successfuly Sign Up'});
    }
  })(req, res, next);
});

app.get('/logout', function(req, res) {
  const email = getUserEmail(req.user);
  req.logout();
  res.redirect('/login');
});

function isLoggedIn(req, res, next) {

    if (!req.user) {
        return res.redirect('/login');
    } else if (!req.user.status.active) {
        req.logout();
        return inactive(req, res);
    }
};

function inactive(req, res) {
    return res.render('404.ejs', {
        msg: "Your account is inactive"
    });
}

function throw404(req, res) {
    res.status = 404;
    res.render('404.ejs', {
        _: require('underscore'),
        user: req.user
    });
}

app.get('/ngapp/data/profile', isLoggedIn, function(req, res) {
  var user = req.user;

  async.parallel({
    req:function(callback) {
      PublisherRequest.getPendingRequestByUserId(String(user._id), callback);
    },
    blacked:function(callback) {
      BlackList.hasUser(req.user._id.toString(), callback);
    }
  }, function(err, result) {
    if (err) {
      res.writeHead(500, {'Content-Type':'application/json'});
      res.write(JSON.stringify({reason:err.message}));
    } else {
      var requestResult = {
        id: String(user._id),
        fullName: user.full_name,
        email: user.getActualEmail(),
        additionalEmail: user.contact_info.additional_email,
        country: user.contact_info.country,
        contactPhone: user.contact_info.contact_phone,
        activePublisher: Boolean(user.status.active_publisher),
        blacklistedPub: !!result.blacked,
```

```
        alreadySentPubRequest:!!result.req,
        accountType: 'newbie',
        holdTime: '',
        advDate: user.advDate,
        pubDate: user.pubDate

      };

      res.writeHead(200, {'Content-Type':'application/json'});
      res.write(JSON.stringify(requestResult));
    }

    res.end();
  });
});

app.get('/ngapp/data/stat/approved/info', isLoggedIn, isActivePublisher,
function(req, res){
  var all_info = [];
  Site.find({user_id: req.user._id}, 'status',  function(err,
publisher_sites){
      if (err){
          res.status(500).send('User not found');
      }
        all_info.push({publisher_sites: publisher_sites});

        AdBlock.find({pub_id: req.user._id}, 'active', function(err, info){
          if (err){
            res.status(500).send('User not found');
          }
            all_info.push({active_blocks: info});
            res.json(all_info);
        });
  });
});

app.get('/ngapp/data/stat/approved/info/ad', isLoggedIn, function(req, res){
  var all_info = [];
  var query = { user_id : String(req.user._id) };
    Ad.find({user_id: req.user._id}, 'active approved new', function(err,
result){
      if (err){
        res.status(404).end();
      }
      all_info.push({adc_active: result});
        Ad.aggregate([
          {$match: query},
          {$group:{_id:'$campaign_id', active : {$max:'$active'} }}
        ]
        ,function(err, result) {
          if (err){
            res.status(404).end();
          }
           all_info.push({cmp_active: result});
           res.json(all_info);
        });
    });
});

app.get('/ngapp/data/campaign/ad/:campaignId', isLoggedIn, function(req,
res){
    Campaign.findById({_id: req.params.campaignId}, 'type', function(err,
result){
      if (err){
```

```
        res.status(500).end();
      }
      res.json(result);
    });
});


app.get('/ngapp/data/campaign/ad/type/:advertId', isLoggedIn, function(req,
res){
    Ad.findById({_id: req.params.advertId}, function(err, advert){
      if (err){
        res.status(500).end();
        log.error(err);
      }
        Campaign.findById({_id: advert.campaign_id}, 'type', function(err,
campaign){
            if (err){
                res.status(500).end();
            }
            res.json(campaign);
        });
    });
});

app.get('/ngapp/data/campaign/type/:campaignId', isLoggedIn, function(req,
res){

 Campaign.findById({ _id: req.params.campaignId }, 'type', function(err,
campaign){
      if (err){
        res.status(500).end();
        log.error(err);
      }
      res.json(campaign);
 });

});

app.get('/ngapp/data/ad/campaigns', isLoggedIn, function(req, res) {
  var timezone = parseInt(req.param('clientTimeZone'));
  if (!isFinite(timezone))timezone = 0;
  var query = {user_id:String(req.user.id)};

  async.parallel({
    campaigns: function(callback) {
      async.waterfall([
        function(callback) {
          Campaign.find({user_id:String(req.user.id), epomId: { $exists:
false }}, callback);
        },
        function(camps, callback) {
          async.each(camps, function(camp, callback) {
            getCampaignBalanceAndCampaignCharge(String(camp._id),
function(err, result) {
              var tempbal = result.balance - result.charge;
              camp.balance = (typeof tempbal === 'number' && tempbal ===
tempbal) ? tempbal : 0;
              camp.advert_charge = (result.advert_charge) ?
result.advert_charge : [];

              callback(null);
            }, String(req.user.id));
          }, function(err) {callback(err, camps)});
        }
```

```
      ], callback);
    },
    data: function(callback) {
      Ad.aggregate([
        {$match:{user_id:String(req.user.id), epom_id: { $exists: false }}},
        {$group:{_id:'$campaign_id', avgBid:{$avg:'$details.bid'},
active:{$max:'$active'}, adverts:{$sum:1}}}
      ], function(err, res) {callback(err, indexById(res));});
    }
  }, function(err, result) {
    if (err)return res.status(500).json({reason:err.message});
    let dataids = _.keys(result.data);
    let theresult = _.map(result.campaigns, function(camp) {
      let campid = String(camp._id);
      let hasdata = _.contains(dataids, campid);

      return {
        campaignId: campid,
        campaignName: camp.campaign_name,
        campaignType: camp.type,
        adsNumber: hasdata ? result.data[campid].adverts : 0,
        avgBid: hasdata ? result.data[campid].avgBid.inttobtc() : '-',
        campaignBalance: camp.balance.inttobtc(),
        campaignStatus: {
          isRunning: hasdata ? result.data[campid].active : false,
          reason: camp.limit.reached ? 'LIMIT_REACHED_PAUSE' : undefined
        }
      };
    });
    res.json({campaigns:theresult});
  });
});

app.post('/ngapp/data/ad/addresponsiveadvertisemnet', isLoggedIn, async
function(req, res) {
  const userid = req.user._id;
  const safeBody = req.body;

  if(!safeBody.campaignId || !safeBody.adType || !safeBody.adType ===
"responsive" || !safeBody.headline
      || !safeBody.description || !safeBody.businessName ||
!safeBody.language || !safeBody.clickUrl || !safeBody.campaignType
      || !safeBody.title || !req.files.logoFile[0] ||
!req.files.backgroundFile[0]) {
    return res.status(400).json({ error : 'Not complete request' });
  }

  let params = {
    user_id         : userid,
    campaign_id     : safeBody.campaignId,
    ad_type         : safeBody.adType,
    details         : {
      title         : safeBody.title,
      description1  : safeBody.description,
      language      : safeBody.language,
      clickurl      : safeBody.clickUrl,
      bid           : (+safeBody.clickPrice).btctoint(),
      headline        : safeBody.headline,
      businessName    : safeBody.businessName,
      logo          : {
        img_id      :
req.files.logoFile[0].path.split(imageFolder.substring(imageFolder.indexOf('/
'), imageFolder.length))[1]
```

```
      },
      background    : {
        img_id      :
req.files.backgroundFile[0].path.split(imageFolder.substring(imageFolder.inde
xOf('/'), imageFolder.length))[1]
      }
    }
  };

  try {
    const campaign = await Campaign.findOne({ _id: safeBody.campaignId });
    let newAd = new Ad(params);
    newAd.cpa = campaign ? campaign.cpa : false;
    newAd = await newAd.save();
    let totalAdsCount = await Ad.find({ approved: false, new: true
}).count();
    woopra.identify(getUserEmail(req.user), getUserData(req.user));
    woopra.track('add_advert', params);
    if (totalAdsCount !== 1){
      return res.json({advertId:String(newAd._id)});
    }
    let user = await User.findById(req.user._id);
    user.mauticSynced = false;
    await user.save();
    res.json({ advertId : String(newAd._id)});
  } catch(error) {
    return res.status(500).json({ error : error.message });
  }

});

app.post('/ngapp/data/ad/addadvert', isLoggedIn, function(req, res) {
  var userid = String(req.user._id);
  var par = req.body;
  var error;

  if (!par.campaignId ||
    !par.adType ||
    !par.title ||
    !par.language ||
    !par.clickUrl ||
    !(+par.clickPrice).btctoint() > 0 ||
    !(par.adType === 'image' || par.adType === 'text') ||
    !((par.adType === 'image' && ((par.imageType === 'static' ||
par.imageType === 'dynamic' || par.imageType === 'flash') &&
req.files.file.length)) ||
    (par.adType === 'text' && (par.adContent && par.adContent.descrStr1 &&
par.adContent.descrStr2 && par.adContent.displayUrl))))
  {error = 'NOT_COMPLETE_REQUEST';}

  if (error) {return res.status(403).json({reason:error});}

  var params = {
    user_id:userid,
    campaign_id:par.campaignId,
    ad_type:par.adType,
    details:{
      language:par.language,
      clickurl:par.clickUrl,
      title: par.title,
      bid: (+par.clickPrice).btctoint()
    }
  };
  if (params.ad_type === 'text') {
```

```
    _.defaults(params.details, {
      displayurl:par.adContent.displayUrl,
      description1:par.adContent.descrStr1,
      description2:par.adContent.descrStr2
    });
  } else if (params.ad_type === 'image') {
    params.details.images = {};
    params.img_type = par.imageType;
    _.each(req.files.file, function(file) {
      var originalName = file.originalname.split('.').splice(0, 1)[0];
      if (!/^\d+x\d+$/.test(originalName))return;
      var sizes = originalName.split('x');
      params.details.images['ad-' + originalName.replace('x', '-')] = {
        img_width:+sizes[0],
        img_height:+sizes[1],

img_id:file.path.split(imageFolder.substring(imageFolder.indexOf('/'),
imageFolder.length))[1]
      };
    });
  }

  async.waterfall([
    function(callback) {
      Campaign.findOne({_id: par.campaignId}, callback);
    },

    function(campaign, callback) {
      var thead = new Ad(params);
      thead.cpa = campaign ? campaign.cpa : false;
      thead.save(function(err, result){
        if (err){
          return callback(err);
        }
        callback(null, result, campaign);
      });
    },

    function(thead, campaign, callback){
      Ad.find({ approved: false, new: true }, function(err, result){
        if (err){
          return callback(err);
        }
        if (!result){
          return callback(null, thead, 0);
        }
        callback(null, thead, result.length, campaign);
      });
    }

  ], function(err, thead, totalAds, campaign) {
    if (err) {
      return res.status(500).json({reason:err.message});
    }

    if (totalAds != 1){
      return res.json({advertId:String(thead._id)});
    }

      User.findById(req.user._id ,function(err, user){
          if (err){
            return res.status(500).end();
          }
          req.user.mauticSynced = false;
```

```
          req.user.save();
          res.json({advertId:String(thead._id)});
      });

  });
});

app.post('/ngapp/data/ad/item/:adid', isLoggedIn, function(req, res) {
  var adid = req.param('adid');
  var bid = (req.body.clickPrice) ?
parseFloat(req.body.clickPrice).btctoint() : null;

  if(isNaN(bid)){
    log.error('bid ' + bid );
    return res.status(500).end();
  }

  async.waterfall([
    function(callback){
      Ad.findOne({ _id: adid }, function(err, result){
          if (err){
            return callback(err);
          }
          if (!result){
            return callback('Ad not found!');
          }
          callback(null, result.campaign_id);
      });
    },
    function(camp_id, callback){
      if (bid === null){
        return callback(null);
      }
      Campaign.findOne({ _id: camp_id }, 'type', function(err, result){
          if (err){
            return callback(err);
          }
          if (result.type === 'native_cpc'){
            if (bid < settings.minBid.cpc){
              return callback('campaign bid cpc to low');
            }
            return callback(null);
          } else {
            if (bid < settings.minBid.cpm){
              return callback('campaign bid cpm to low');
            }
            return callback(null);
          }
      });
    },
    function(callback) {
      var changes = { modified:adModTime.get() };
      var edit = false;
      var onlyChangeBid = false;
      var adContentLength = (req.body.adContent) ?
Object.keys(req.body.adContent).length : 0;

      if (req.body.clickPrice) {
        changes['details.bid'] = bid;
        edit = true;
      }

      if (req.body.clickUrl) {
        changes['details.clickurl'] = req.body.clickUrl;
```

```
        edit = true;
      }

      if (req.body.title) {
        changes['details.title'] = req.body.title;
        edit = true;
      }

      if (req.body.adContent) {
        var reqcont = req.body.adContent;

        if (reqcont.descriptionStr1) {
          changes['details.description1'] = reqcont.descriptionStr1;
          edit = true;
        }

        if (reqcont.descriptionStr2) {
          changes['details.description2'] = reqcont.descriptionStr2;
          edit = true;
        }

        if (reqcont.displayUrl) {
          changes['details.displayurl'] = reqcont.displayUrl;
          edit = true;
        }
      }

      if (req.body.clickPrice && !req.body.clickUrl && !req.body.title &&
adContentLength == 0){
          onlyChangeBid = true;
      }

      if (edit && !onlyChangeBid) {
        // changes.active = false;
        changes.approved = null;
        changes.new = false;
        changes.deny_reason = 'MODIFIED';
      }

      Ad.update({_id:adid, user_id:String(req.user._id)}, changes, {
multi:false }, function(err){
        if (err){
          return callback(err);
        }
        callback(null);
      });
    },

    function(callback){
      Ad.find({ approved: null, new: false, deny_reason: 'MODIFIED' },
function(err, result){
        if (err){
          return callback(err);
        }
        if (!result){
          return callback(null, 0);
        }
        callback(null, result.length);
      });
    }

  ], function(err, totalAds) {
    if (err) {
      log.error(err);
```

```
      return res.status(500).end();
    }

    if (totalAds == 0){
      return res.end();
    }
    if (totalAds != 1){
      return res.end();
    }

    User.findById(req.user._id ,function(err, user){
          if (err){
            return res.status(500).end();
          }
          res.end();
    });

  });
});
```

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2018 р.

ВЕБ-ПЛАТФОРМА ДЛЯ РОЗМІЩЕННЯ РЕКЛАМИ. СЕРВЕРНА ЧАСТИНА

Програма та методика тестування

ДП.045440-04-51

"ПОГОДЖЕНО"

Керівник проекту:

_____ Н.А. Рибачок

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.О. Дяченко

2018

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Веб-платформа для розміщення реклами являє собою веб-додаток, створений на платформі Node.js (мова програмування JavaScript), з використанням архітектурного шаблону клієнт-сервер.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) відповідність програмного інтерфейсу платформи вимогам до функціональних можливостей веб-платформи;
2) наявність доступу до платформи з будь-якого сучасного браузера;
4) забезпечення належного рівня безпеки даних;
3) наявність обробки критичних ситуацій;
4) зручність роботи з веб-платформою;

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Black Box Testing. Перевіряється безпосередньо програмний продукт на відповідність функціональним вимогам з точки зору зовнішнього світу, без урахування знань про внутрішнє представлення системи. Тестування відбувається на рівні системного тестування.

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування граничних значень;
4) тестування за допомогою припущення про помилку;
5) тестування програмного інтерфейсу платформи.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність серверної частини веб-платформи для розміщення реклами перевіряється шляхом:

1) динамічного ручного тестування – введенням коректних, граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування веб-платформи у різних веб- браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування швидкості відповіді на запити;
8) тестування програмного інтерфейсу платформи.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

"ЗАТВЕРДЖЕНО"

Науковий керівник кафедри

_____ І.А. Дичка

"___" _____ 2019 р.

ВЕБ-ПЛАТФОРМА ДЛЯ РОЗМІЩЕННЯ РЕКЛАМИ. СЕРВЕРНА ЧАСТИНА

Керівництво користувача

ДП.045440-05-34

"ПОГОДЖЕНО"

Керівник проекту:

_____ Н.А. Рибачок

Нормоконтроль:

_____ М.В. Онай

Виконавець:

_____ Д.О. Дяченко

2019

ЗМІСТ

1. Опис структури веб-платформи

Веб-платформа для розміщення реклами складається із статичних сторінок та динамічних сторінок, вміст яких формується динамічно. Веб-платформа виконана англійською мовою.

Статичні сторінки веб-платформи:

– початкова сторінка входу до системи;

– сторінка реєстрації у системі.

Динамічні сторінки веб-платформи:

– сторінка дошки користувача;

– сторінка рекламодавця;

– сторінка вебмайстра;

– сторінка списку кампаній;

– сторінка списку рекламних блоків;

– сторінка кампанії;

– сторінка рекламного банера;

– особиста сторінка користувача;

– сторінка поповнення балансу акаунта;

– сторінка поповнення балансу кампанії;

– сторінка створення рекламного блоку;

– сторінка створення рекламної кампанії;

– сторінка створення рекламного банера.

Майже кожна сторінка містить посилання на минулу сторінку веб-платформи або наявні елементи, за допомогою яких користувач перейде до наступної сторінки.

2. Опис інтерфейсу сторінок платформи

Початкова сторінка входу до системи.

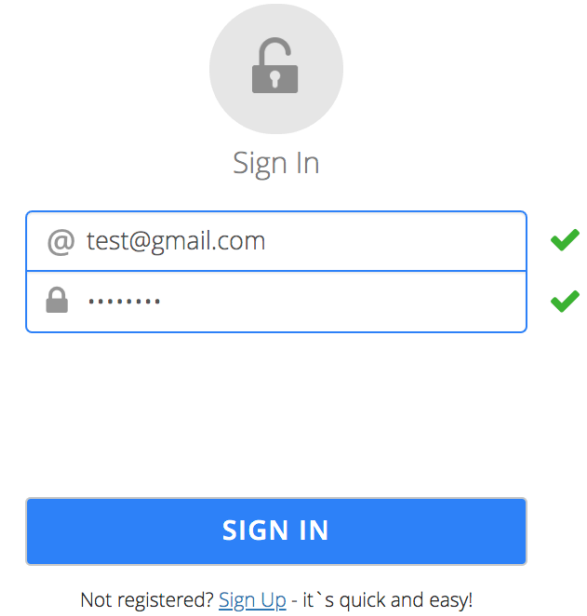


Рис. 1. Початкова сторінка входу до системи

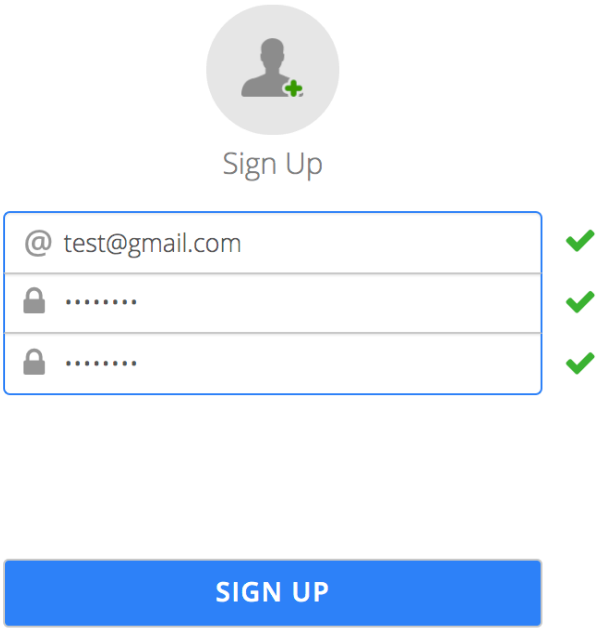


Рис. 2. Сторінка реєстрації

Сторінки для авторизованого користувача.

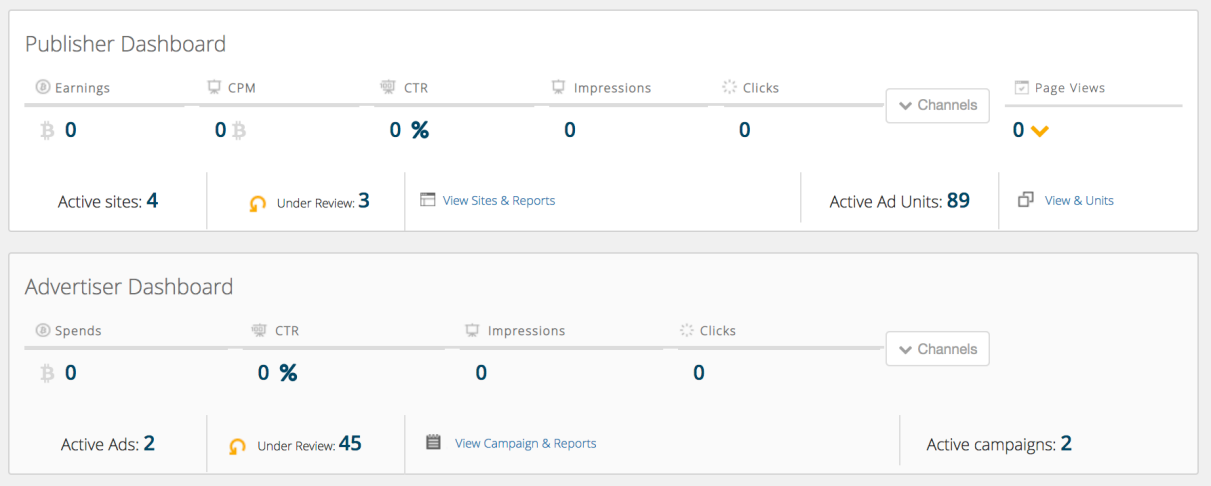


Рис. 3. Дошка користувача

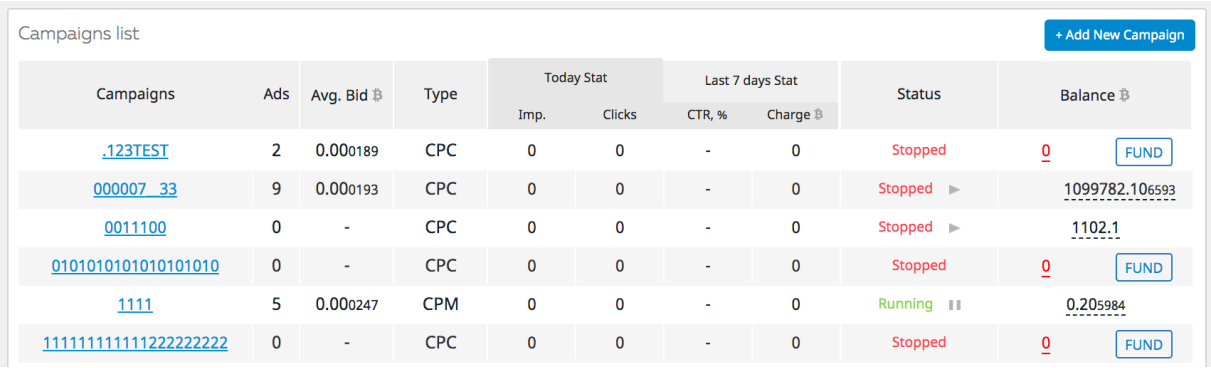


Рис. 4. Дошка рекламодавця, список кампаній

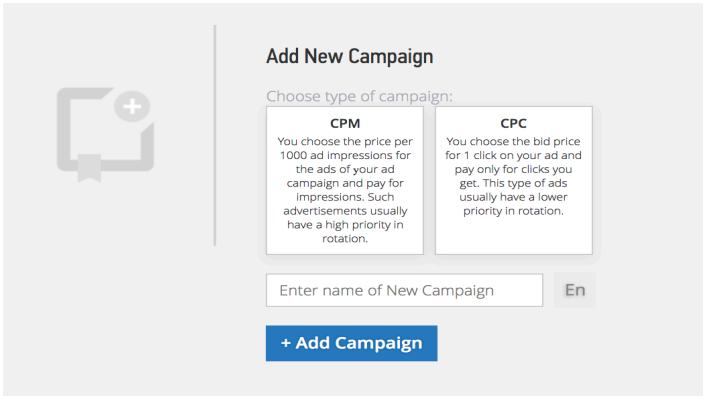


Рис. 5. Створення кампанії

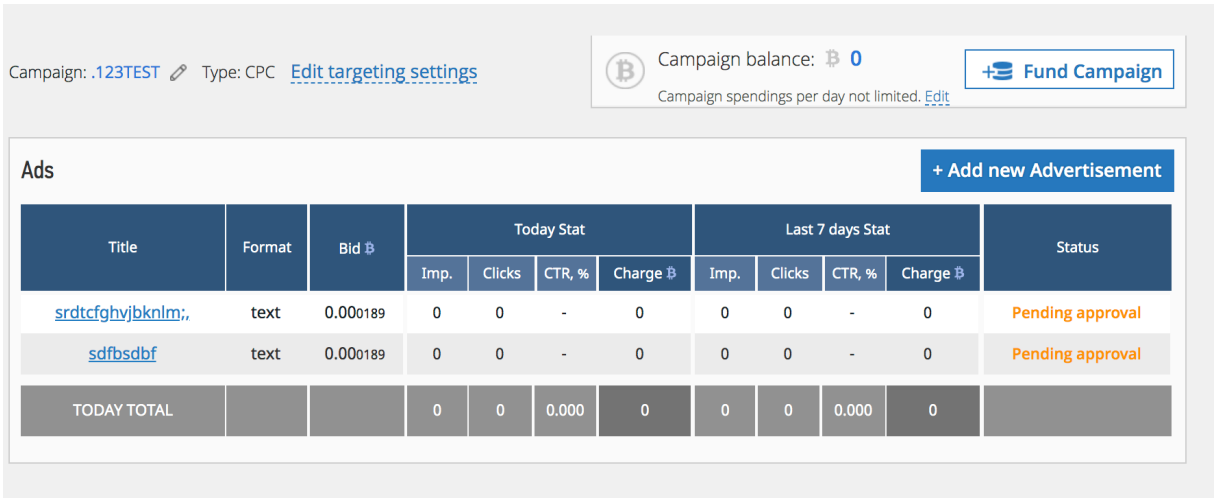


Рис. 6. Частина сторінки кампанії

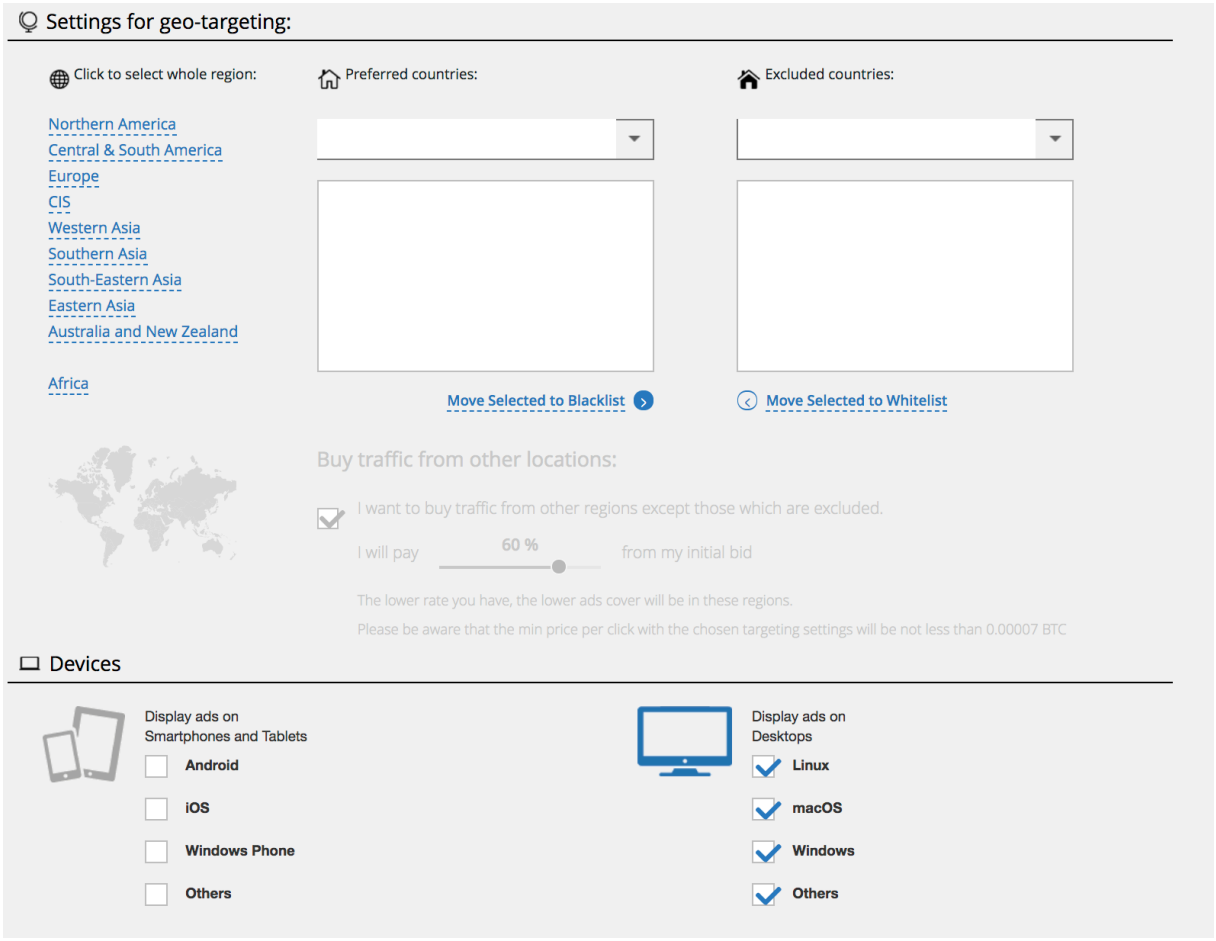


Рис. 7. Налаштування таргетингу кампанії

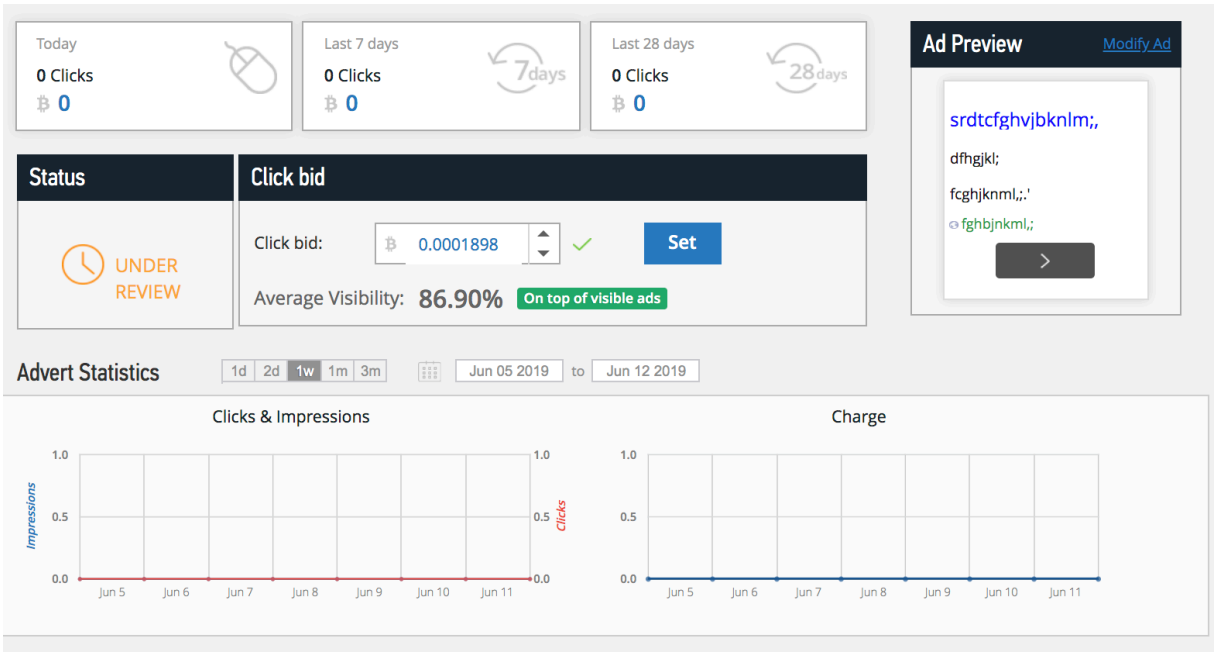


Рис. 8. Сторінка рекламного блоку

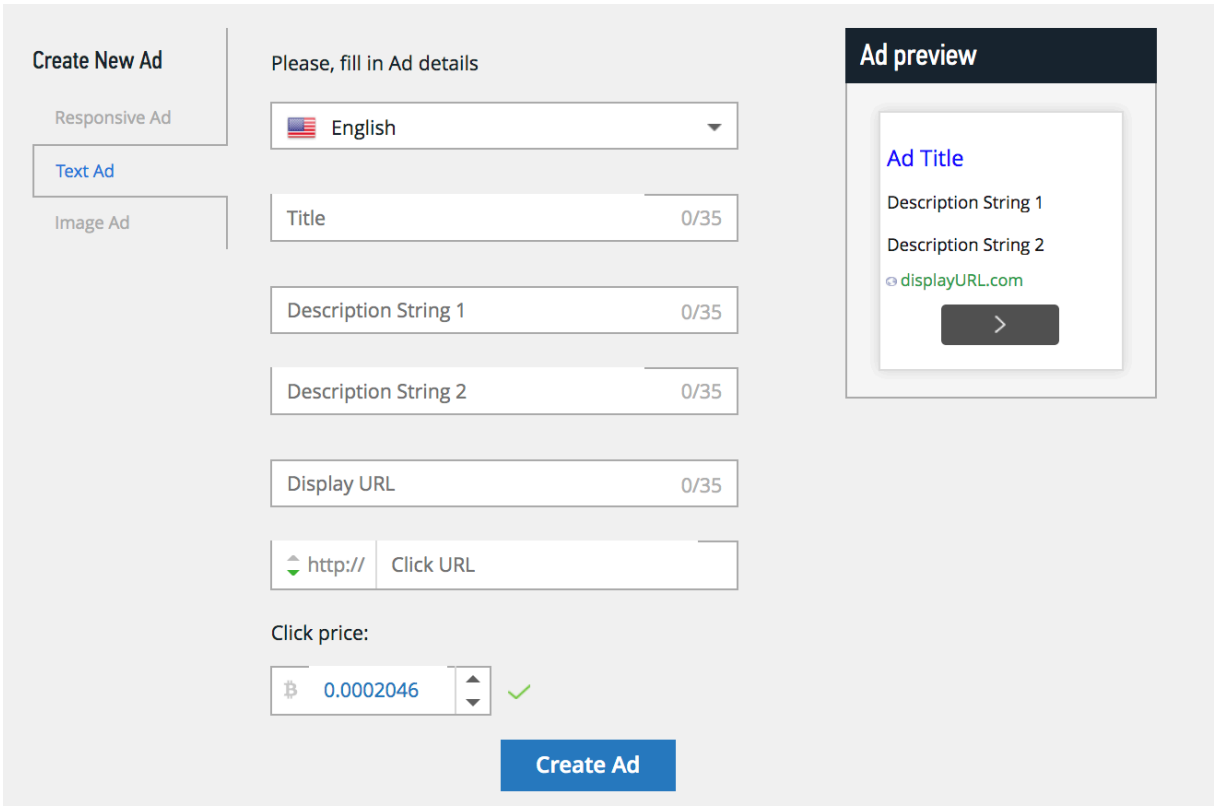


Рис. 9. Створення рекламного банера

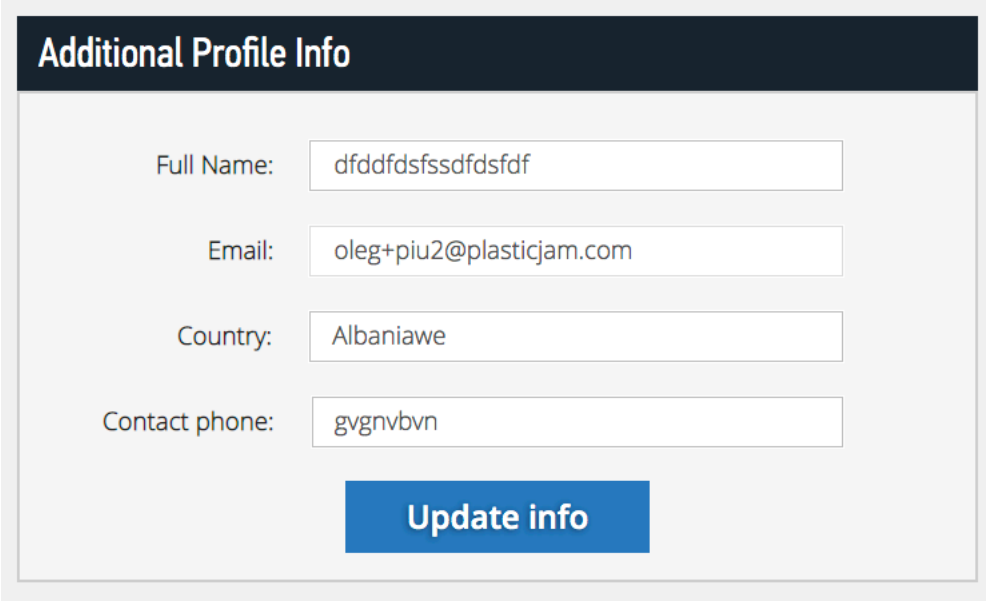


Рис. 10. Сторінка інформації про користувача

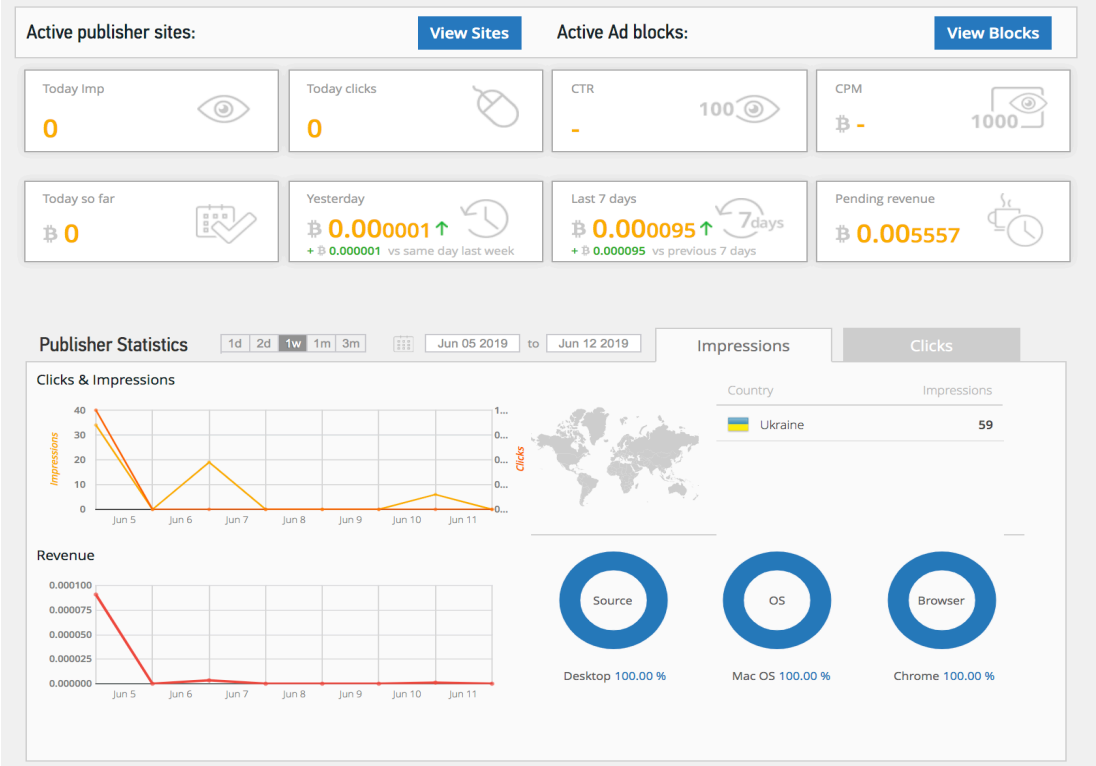


Рис. 11. Дошка вебмайстра

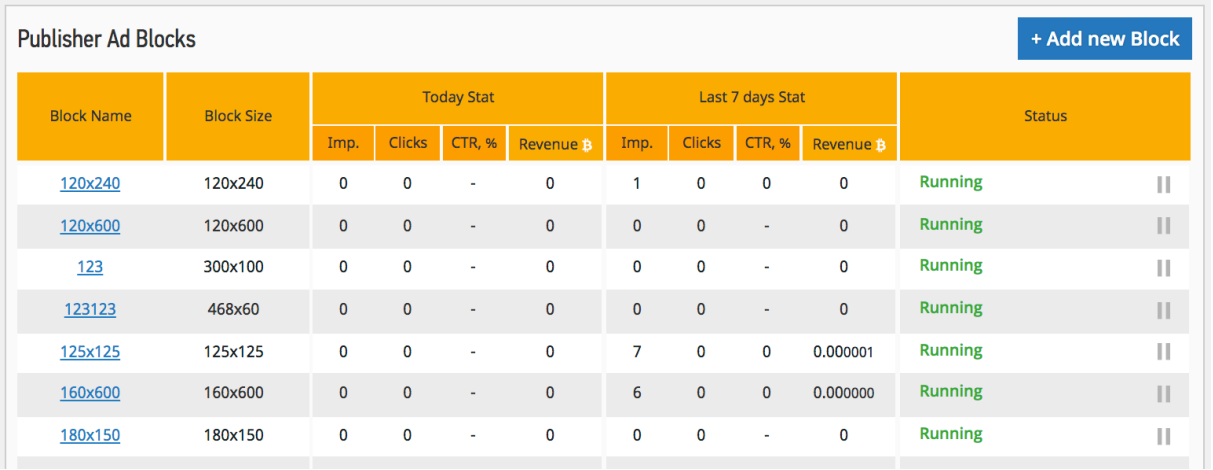


Рис. 12. Сторінка списку рекламних блоків

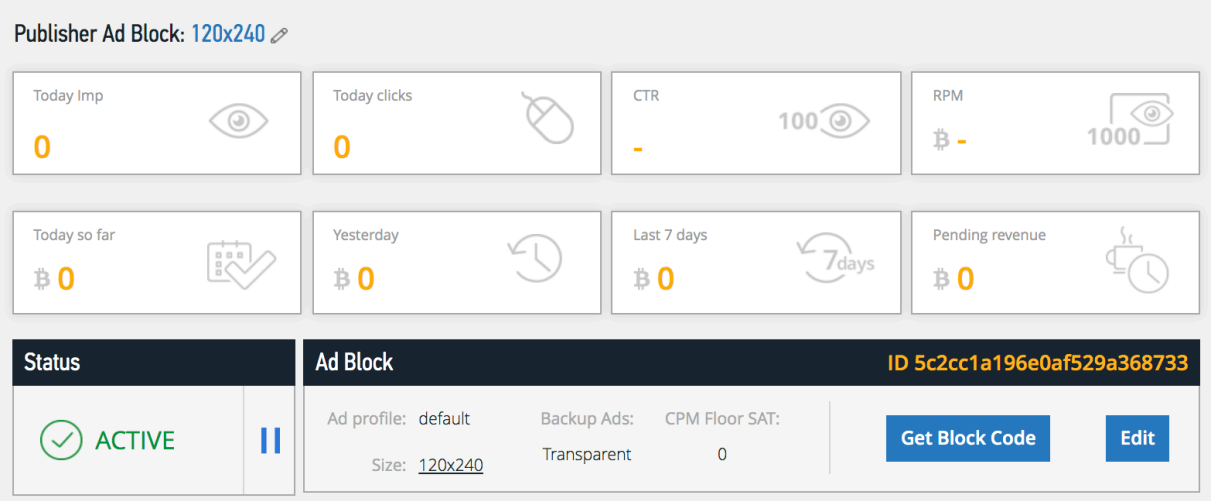


Рис. 13. Сторінка рекламного блоку

3. Процедура реєстрації та авторизації користувача

Процедура реєстрації.

Для того щоб зареєструватись користувач має ввести наступні дані – e-mail, пароль та повторити пароль. Введення даних до всіх полів оброблено згідно необхідної валідації. Наприклад, якщо користувач ввів неправильний e-mail, то виникає помилка (рис. 14).

Після процедури реєстрації користувач автоматично не авторизується в системі.

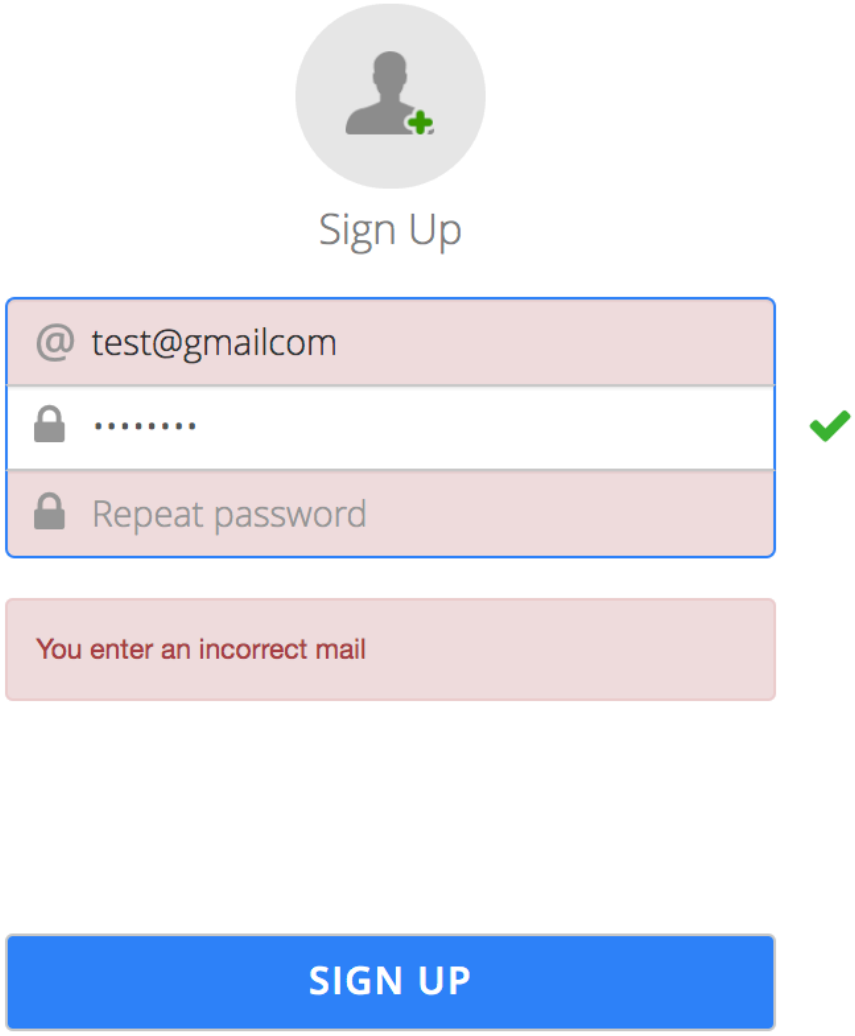


Рис. 14. Помилка при введені некоректних даних

Процедура авторизації.

Для того, щоб авторизуватись у системі користувач повинен ввести e-mail та пароль, який він вводив при реєстрації. Введення даних до всіх полів

оброблено згідно необхідної валідації. Помилки оброблено аналогічно до полів на сторінці «Реєстрація».

4. Процедура створення рекламної кампанії та рекламного банера

Після авторизації користувач переходить до сторінки головної дошки (рис. 3).

Створення рекламної кампанії.

Для того, щоб створити рекламну кампанію треба перейти до дошки рекламодавці та натиснути кнопку Add new campaign (рис. 4). Після цього потрібно заповнити поле для назви кампанії, обрати тип кампанії та натиснути кнопку Add campaign (рис. 5).

Додавання рекламного банера.

Для того, щоб створити рекламний банер, потрібно спочатку створити рекламну кампанію, за перейти на сторінку рекламної кампанії (рис. 6). Після цього натиснути кнопку Add new advertisement. Відкриється сторінка створення рекламного банера (рис. 9). Потрібно обрати тип банера (на панелі зліва) та заповнити усі текстові поля. Після цього натиснути кнопку Create ad.

5. Процедура реєстрації сайту та додавання рекламного блоку

Після авторизації користувач який не є вебмайстром може заповнити заявку на отримання статусу вебмайстра, для цього потрібно зареєструвати свій сайт (рис. 15). Потрібно заповнити усі поля та натиснути кнопку Add publisher Site. Потім він з’явиться у списку сайтів вебмайстра (рис. 16).

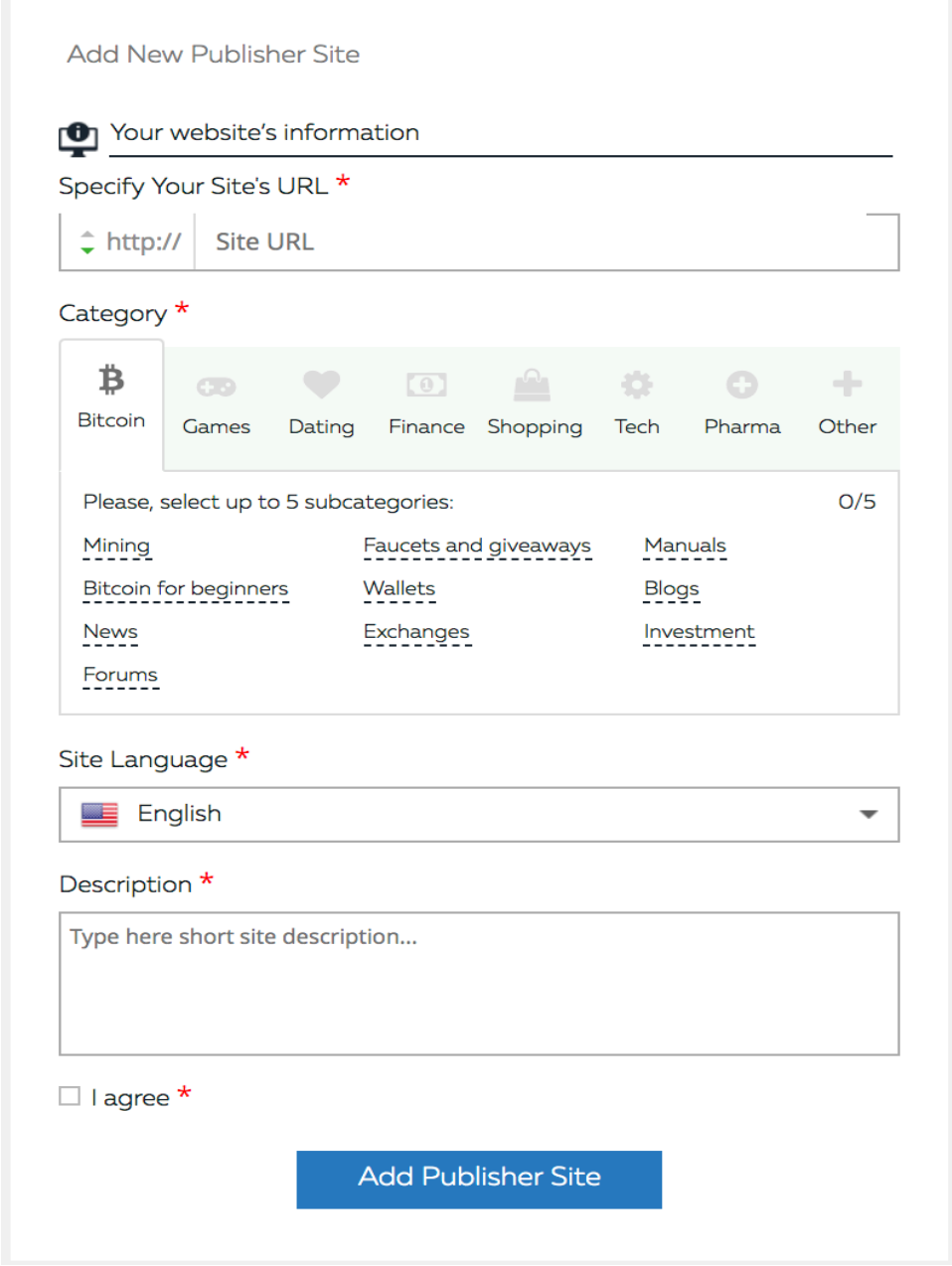


Рис. 15. Реєстрація сайта

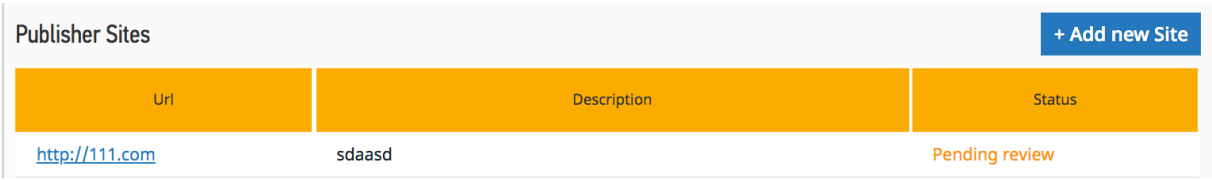


Рис. 16. Список сайтів вебмайстра

Створення рекламного блоку.

Після реєстрації першого сайту на сторінці рекламних блоків можна створити рекламний блок натиснувши кнопку Add new Block (рис. 12). Після

цього потрібно заповнити усі необхідні поля та натиснути кнопку Add Block. Він з’явиться у списку блоків (рис. 12).